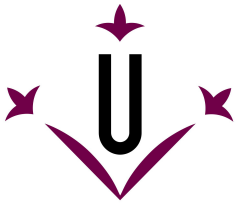
Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes
Treball de final de carrera

Disseny i implementació de l'aplicació Protein-MetReS



Universitat de Lleida

Autors: Adrià Ramírez Trallero
Silvia Masot Esteve
Directors:Francesc Solsona Tehas
Anabel Usié Chimenos
Setembre 2011

RESUM

Actualment, dins del món de la biomedicina i bioquímica, i pel que fa a investigadors i grups de recerca, sorgeix la necessitat d'haver de treballar amb diferents i nous programes o eines que facilitin la tasca que desenvolupen els biòlegs i bionformàtics en el camp de la reconstrucció de xarxes metabòliques. Són moltes i variades les aplicacions informàtiques que es poden trobar a l'abast. No obstant, el present projecte sorgeix de la idea de facilitar la tasca als experts mitjançant la creació d'una aplicació informàtica a mida. D'aquí neix el nostre projecte final de carrera, una aplicació capaç d'oferir a l'usuari una interfície gràfica per la qual podrà navegar còmodament, realitzant diferents tasques com són la cerca d'estructures proteiques, la creació i/o predicció de models estructurals i la realització d'un procés anomenat "Docking".

Per tant, el que s'ha dut a terme ha estat l'anàlisi, disseny i implementació d'una aplicació, anomenada Protein-MetReS, integrada dins del projecte MetReS. Aquesta aplicació ha estat desenvolupada sota la tecnologia Java que, a través de servidors web i servidors localment instal·lats, a partir de la selecció d'un organisme realitza cerques sobre bases de dades que contenen estructures de proteïnes conegudes i, en el seu defecte, si aquestes no existeixen, crea el model tridimensional en base als homòlegs d'estructura coneguda, per després sotmetre'ls a un procés d'acoblament molecular, o més conegut, com a docking.

El present document conté la memòria de tot el treball realitzat durant el desenvolupament de l'aplicació. En la finalització d'aquest projecte s'ha arribat a la conclusió de que l'aplicació ha demostrat funcionar correctament en la majoria de casos. Tot i això necessita d'algunes millores per acabar sent una aplicació òptima per a investigadors i grups de recerca, ja que hi ha execucions que triguen molt temps en executar-se. A més a més, alguns dels programes que havien de formar part del projecte, com són els que funcionen de forma local, han quedat pendents d'incorporar-se.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

Aquest projecte parteix arrel d'un altre projecte sobre una de les primeres aplicacions en funcionament dins del servidor biomèdic anomenada Biblio-MetReS, la qual va ser desenvolupada per Anabel Usié Chimenos i Xavier Faus Torà, ambdós membres pertanyents a grups d'investigació de la Universitat de Lleida.

En segon lloc, destacar la participació de David Terés Carrillo, el qual va desenvolupar la instal·lació i configuració del servidor biomèdic. La configuració del servidor ha fet possible el desplegament de noves aplicacions com és en aquest cas l'aplicació Protein-MetReS.

Els projectes esmentats amb anterioritat no es detallaran a continuació ja que no és el propòsit del present, però si s'exposen en el seu Projecte Final de Màster pel que fa als dos primers membres anomenats i Projecte Final de Carrera pel que respecta a David Terés.

Aquest projecte s'ha dut a terme per cobrir les necessitats que han aparegut en els grups de recerca en el camp de la biomedicina. És per aquest motiu que ha sorgit la idea de dissenyar i implementar una aplicació capaç d'oferir la possibilitat als usuaris de realitzar cerques d'estructures proteiques, creació de models estructurals si aquestes no són conegudes i com a procés molecular, l'execució d'un procés d'acoblament.

L'aplicació interactua amb l'usuari per mitjà d'una interfície gràfica. A través d'aquesta interfície serà possible realitzar les diferents tasques que ens permet realitzar l'aplicació, com són la cerca de models estructurals, la creació i/o predicció d'aquestos i per últim la realització d'un procés de docking. Tot això serà possible gràcies a la comunicació que s'estableix entre l'aplicació i el servidor biomèdic, el qual incorpora la implementació d'un web service que farà possible la interacció amb aplicacions externs i la base de dades del servidor. El servidor biomèdic es comunica via internet i mitjançant la tecnologia SOAP amb l'aplicació Protein-Metres.

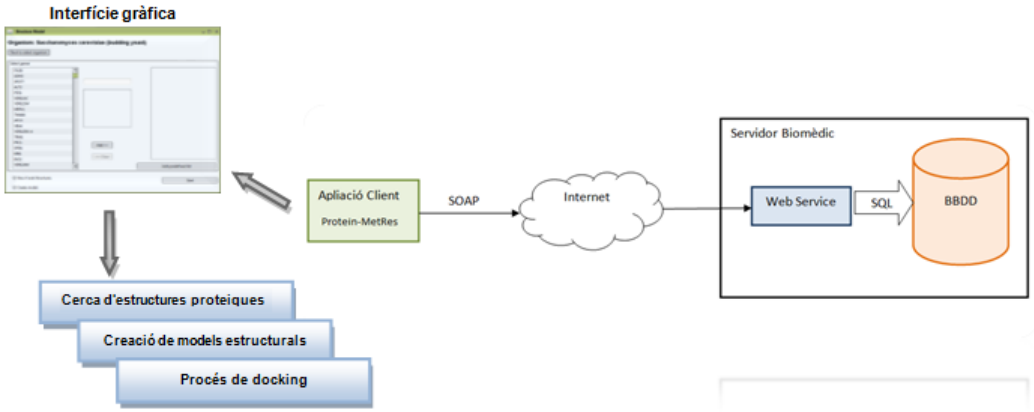


Figura 1.1: Diagrama Protein-MetReS

1.1 Motivació

La motivació principal ha estat la necessitat d'integrar una eina de docking dins del projecte global MetReS, on ja s'hi ha desenvolupat una eina de reconstrucció de xarxes metabòliques anomenada Biblio-MetReS.

Una motivació addicional ha estat la necessitat de generar i desenvolupar una eina útil pels grups de recerca i/o investigació en els camps de la biologia, biomedicina i bioinformàtica.

A nivell personal, una altra motivació ha estat el fet d'endinsar-nos dins d'un camp desconegut per a nosaltres com ha estat el camp de la biologia. El fet de tenir que assumir nous conceptes i coneixements sobre un camp que no és el propi, sempre suposa un esforç afegit però, al mateix temps, una motivació.

Per últim, el fet de tenir que programar en un llenguatge fins ara desconegut per nosaltres, com és el llenguatge Java, ha suposat marcar-nos un nou objectiu personal, com és arribar a concloure un projecte d'aquest tipus, amb l'afegit d'haver de tenir que desenvolupar-nos dins d'una interfície gràfica que fins al moment tampoc ens havíem trobat mai.

1.2 Objectius

L'objectiu principal d'aquest Treball Final de Carrera ha estat dissenyar i implementar una aplicació informàtica de docking (anomenada Protein-MetReS) de proteïnes útils per als experts en diferents camps de biologia. Com a objectius específics cal destacar:

- L'eina ha de ser accessible per múltiples usuaris via web.
- Capaç de complir amb totes les seves funcions, com s'ha comentat anteriorment, de

donar la possibilitat a cada usuari de realitzar tant la creació de models estructurals com la cerca d'estructures moleculars.

- L'aplicació ha de proporcionar una interfície gràfica usable i intuïtiva, facilitant així la tasca a l'usuari final.
- Incorporar utilitats i eines de docking en el servidor del projecte MetReS. En aquest servidor també està allotjada l'aplicació Biblio-MetReS.
- Construcció d'interfícies d'accés al servidor mitjançant la tecnologia de Web Services.

A nivell personal, destacar com objectiu el repte d'aprendre un nou llenguatge de programació, amb noves tecnologies, i l'adquisició de nous coneixements que fins al moment i al llarg de la carrera no havien estat assolits. A més a més, la realització del projecte sempre posa a prova alguns dels coneixements adquirits prèviament.

1.3 Conceptes previs

En aquest apartat s'explica de forma resumida els diferents conceptes que s'han utilitzat per la realització d'aquest projecte.

1.3.1 Conceptes Biològics

- **Xarxa de gens i/o proteïnes**: Conjunt de gens que participen en un mateix procés biològic i que interaccionen d'alguna manera entre ells, ja sigui funcional o físicament.
- **Gen**: És la seqüència d'àcid desoxiribonucleic (ADN) que constitueix la unitat funcional per a la transmissió de caràcters hereditaris. Un gen és una seqüència lineal de nucleòtids que conté la informació necessària per sintetitzar una macromolècula amb funció cel·lular específica. El gen, com a unitat d'emmagatzemament d'informació genètica, té la funció de transmetre l'herència a la descendència. El conjunt de gens d'una mateixa espècie es coneix com a genoma, mentre que la ciència que ho estudia rep el nom de genètica.
- **Proteïna**: Les proteïnes són cadenes d'aminoàcids que es pleguen adquirint una estructura tridimensional que els permet dur a terme milers de funcions. Les proteïnes estan codificades en el material genètic de cada organisme, on s'especifica la seva seqüència d'aminoàcids, i després són sintetitzades pels ribosomes. Les proteïnes posseeixen totes una mateixa estructura química central, que consisteix en una cadena lineal d'aminoàcids. El que fa diferent a una proteïna d'una altra és la seqüència d'aminoàcids de que està feta. A tal seqüència es coneix com l'estructura primària de la proteïna.

- **Seqüència d'ADN**: És una successió de lletres que representen l'estructura primària d'una molècula real o hipotètica d'ADN, amb la capacitat de transportar informació. Les possibles lletres són A, C, G, i T, que simbolitzen les quatre subunitats de nucleòtids d'una banda ADN - adenina, citosina, guanina, timina, que són bases de tipus covalent lligades a cadenes fosfòriques. Normalment, les seqüències es representen mitjançant una cadena contínua de nucleòtids. Exemple de seqüència:

MIVKVKTLTGKEISVELKESDLVYHIKELLEEKEGIPPSQQRLIFQGKQIDDKLTVTDAHLVEGMQLHLVLTLRGGN

- **Estructura proteica**: L'estructura de les proteïnes reuneix les propietats de disposició en l'espai de les molècules de proteïna que provenen de la seva seqüència d'aminoàcids, les característiques físiques del seu entorn i la presència de compostos, simples o complexos, que les estabilitzin i/o condueixen a un plegament específic. Al voltant del 90% de les estructures de les proteïnes disponibles en el Banc de Dades de Proteïnes (PDB), han estat determinades per cristal·lografia de rajos X. Més endavant es pot veure en que consisteix aquest procés de cristal·lització.

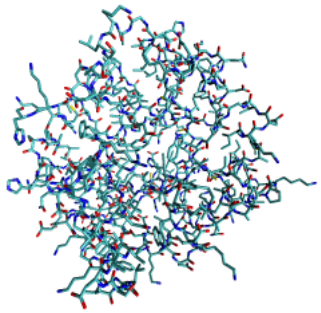


Figura 1.2: Estructura primària de la proteïna

- **Model tridimensional**: El model tridimensional d'una proteïna és el resultat d'una predicció sobre la conformació dels seus àtoms, ja que no presenta una estructura coneguda i disponible en el PDB. Mitjançant un modelatge per homologia és possible realitzar una predicció de quina estructura conformarà una proteïna basant-se en altres estructures de característiques similars. Més endavant es pot veure en què consisteix el procés de modelatge per homologia.

- **Docking**: És la unió de dues molècules, ja siguin proteïnes o no, una rep el nom de ligand i l'altre rep el nom de receptor, per tal de formar un complex estable. Aquest procés d'unió consisteix en descobrir quina és la millor posició del ligand per al seu receptor. També és conegut amb el nom d'acoblament molecular.

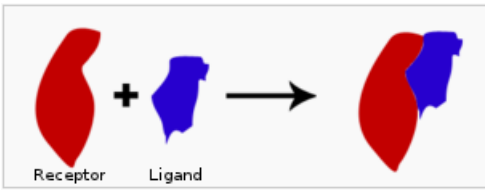


Figura 1.3: Diagrama de docking

  - **Receptor**: És la molècula que rep al ligand.
  - **Ligand**: És la parella complementària que s'enllaça al receptor.

- **Determinació d'estructures proteiques per cristal·lografia:** Aquest mètode permet mesurar la densitat de distribució dels electrons de la proteïna en les 3 dimensions (en l'estat de cristal·lització), la qual cosa permet obtenir les coordenades 3D de tots els àtoms per determinar la seva posició amb certesa. El procés consisteix en fer passar un feix de rajos X a través d'un cristall de la substància subjecta a estudi. El feix s'escindeix en diverses adreces a causa de la simetria de l'agrupació d'àtoms i, per difracció, dóna lloc a un patró d'intensitats que pot interpretar-se segons la ubicació dels àtoms en el cristall.

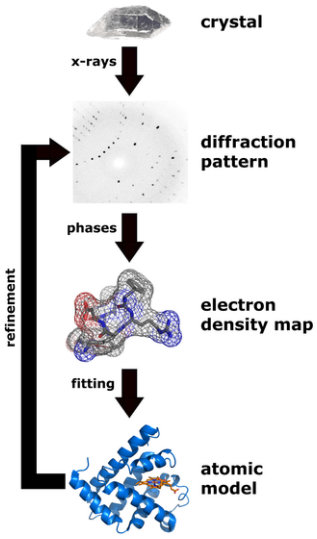


Figura 1.4: Determinació estructura per cristal·lografia de rajos X

- **Modelatge d'estructures proteiques per homologia:** Mètode que es basa en la suposició raonable de que dues proteïnes homòlogues compartiran unes estructures molt similars. Aquesta suposició d'aproximació arrela en el fet que totes les parelles de proteïnes que presenten una identitat de seqüència major al 30% tenen estructura

tridimensional similar. D'aquesta manera es pot construir el model tridimensional d'una proteïna d'estructura desconeguda, partint de la similitud de seqüència amb proteïnes d'estructura coneguda. Aquest procés consta bàsicament de quatre etapes que es poden veure a continuació:

  - *Selecció de plantilla*: consisteix en trobar les estructures principals i serveix com a base per al procés de modelatge. Aquest pas consisteix en la cerca al PDB per seleccionar aquelles proteïnes homòlogues. Aquesta cerca es pot dur a terme mitjançant qualsevol mètode d'alineament de parells. Es recomana utilitzar només aquella estructura amb el percentatge de similitud més alt.
  - *Alineament de seqüències*: un cop s'ha identificat la seqüència amb major similitud, es duu a terme un reajustament. Per realitzar aquest procés s'utilitza un algoritme d'alineament per obtenir una adaptació òptima entre les seqüències. Aquest pas és el més crític ja que la construcció del model es realitzarà envers aquest alineament.
  - *Creació del model*: una vegada es té l'alineament òptim, existeixen diverses aproximacions per construir les coordenades espacials de la seqüència des de l'alineament realitzat. Per exemple tenim ProModII en el servidor SWISS-MODEL.
  - *Avaluació del model*: La informació que es pot obtenir del model depèn de la seva qualitat, de manera que és molt important poder avaluar-la. Existeixen moltes proves que es poden realitzar sobre un model que inclou comprovacions estèriques, químiques, etc.

- **Protein Data Bank:** És una base de dades d'estructures de proteïnes i àcids nucleics en 3D. Aquestes dades, generalment obtingudes per cristal·lografia de rajos X o ressonància magnètica nuclear, són enviades per biòlegs i bioquímics de tot el món. Estan sota el domini públic i poden ser usades lliurement. L'adreça web del servidor és:
  http://www.pdb.org/pdb/home/home.do

- **SwissModel Repository:** És una base de dades on s'estableixen comparacions entre models de proteïnes estructurades en 3D i actualment conté models tridimensionals per a seqüències, alimentat per la base de coneixements de UnitPro, repositori central de dades sobre proteïnes que proporciona recursos de forma lliure a la comunitat científica. L'adreça web del servidor és:
  http://www.proteinmodelportal.org/?aid=queryModellingInteractiveBySeq&zid=async

- **3DJIGSAW:** És un servidor que crea models tridimensionals de les proteïnes sobre la base d'homòlegs d'estructura coneguda. L'adreça web del servidor és:
  http://bmm.cancerresearchuk.org/~3djigsaw/

- **Phyre**: És un servidor de reconeixement automàtic sobre homologia de proteïnes. Prediu l'estructura de la seqüència proteica. L'adreça web del servidor és:
  http://www.sbg.bio.ic.ac.uk/~phyre/

1.3.2 Web Services

Figura 1.5: Web Services

Un servei Web (Web Service en anglès) és un sistema software que té com a objectiu donar suport a la comunicació i interacció màquina-a-màquina dins d'una xarxa. Aquest servei web es comunica mitjançant una interfície en format WSDL (Web Service Definition Language). Una altra forma d'interactuar dels sistemes amb el servei web és utilitzant missatges SOAP (Simple Object Access Protocol) a través de plataformes HTTP (HyperText Transfer Protocol) configurades mitjançant fitxers XML (Extesible Markup Language) en conjunció amb altres estàndards del disseny Web.

Existeix una branca dels serveis web anomenada "Big Web Services", la qual engloba tots aquells serveis web que interactuen amb plataformes dissenyades amb llenguatges de caire general, com Java, Python o C#. Aquests serveis web proporcionen accés a unes funcions concretes que són cridades per les aplicacions externes (anomenades aplicacions client) mitjançant missatges XML utilitzant patrons SOAP. Aquestes funcions ofertades es descriuen en el fitxer WSDL al qual tenen accés les aplicacions client. L'ús d'aquest disseny permet al servei web estalviar-se l'ús dels estàndards de SOAP, però obliga a l'aplicació client a utilitzar un framework específic (com ara Spring o Apache Axis2) que s'encarregui de la generació automàtica de codi per a la codificació i comunicació entre els dos sistemes.

1.3.3 Tomcat

Figura 1.6: Tomcat

Apache Tomcat és un contenidor de servlets[1] desenvolupat per "Apache Software Foundation". Tomcat implementa les especificacions de servlet i de Java Server Pages (JSP) de Sun Microsystems, proporcionant un entorn per al codi Java a executar en cooperació amb

[1] *Servlet: petita aplicació Java que s'executa en un servidor web i que s'envia a l'usuari juntament amb una pàgina web a fi de realitzar determinades funcions, tals com l'accés a bases de dades o la personalització d'aquestes pàgines web

un servidor web. Aquest afegeix eines per a la configuració i el manteniment, però també pot ser configurat editant els fitxers de configuració que normalment són en format XML. Tomcat inclou el seu propi servidor HTTP, per això també se'l considera un servidor web independent.

Tomcat és desenvolupat i mantingut per membres d' "Apache Software Foundation" i voluntaris independents. Els usuaris disposen de lliure accés al seu codi font i a la seva forma binària en els termes establerts en l'Apache License. Les primeres distribucions de Tomcat van ser les versions 3.0.x. Les versions més recents són les 6.x, que implementen les especificacions de Servlet 2.4 i de JSP 2.0. Les versions 4.0 i posteriors, utilitzen el contenidor de servlets "Catalina" internament.

1.3.4 Java

Figura 1.7: Java

Java és un llenguatge de programació orientat a objectes, desenvolupat per Sun Microsystems a principis dels anys 90. El llenguatge en si mateix pren molta de la seva sintaxi de C i C++, però té un model d'objectes més simple i elimina eines de baix nivell, que solen induir a molts errors, com la manipulació directa de punters o memòria.

Les aplicacions Java estan típicament compilades en un bytecode, encara que la compilació en codi màquina natiu també és possible. En el temps d'execució, el bytecode és normalment interpretat o compilat a codi natiu per a l'execució, encara que l'execució directa per maquinari del bytecode per un processador Java també és possible.

La implementació original i de referència del compilador, la màquina virtual i les biblioteques de classes de Java van ser desenvolupats per Sun Microsystems en 1995. Des de llavors, Sun ha controlat les especificacions, el desenvolupament i evolució del llenguatge a través del Java Community Process, tot i que d'altres han desenvolupat també implementacions alternatives d'aquestes tecnologies de Sun, algunes fins i tot sota llicències de programari lliure.

Entre desembre de 2006 i maig de 2007, Sun Microsystems va alliberar la major part de les seves tecnologies Java sota la llicència GNU GPL, d'acord amb les especificacions del Java Community Process, de tal forma que pràcticament tot el Java de Sun és ara

programari lliure (encara que la biblioteca de classes de Sun que es requereix per executar els programes Java encara no ho és).

1.3.5 InfoDebug Web

Per la depuració del codi Web s'ha utilitzat l'extensió Firebug del navegador Firefox. Aquesta extensió ens permet veure el funcionament intern dels llocs webs, la qual cosa ens permet analitzar el codi amb més detall. En l'apartat 2.2.3 es pot observar el funcionament d'aquesta extensió.

1.3.6 Estructura del treball

Al capítol 1 podeu observar una breu explicació dels diferents conceptes previs que hem considerat claus per tal de disposar d'una bona comprensió d'aquesta memòria, a més a més, els motius que han motivat a la realització de la mateixa i els objectius a assolir.

Al capítol 2 s'explica tot el desenvolupament de l'aplicació. En quina plataforma s'ha desenvolupat, quins servidors actuen en l'aplicació, com es realitzen les peticions als servidors, flux d'execució de la mateixa, operacions que es poden dur a terme i la forma de fer-ho entre altres aspectes a considerar en la descripció de l'aplicació.

Al capítol 3 es mostren diferents casos pràctics per cada operació que ens permetrà realitzar l'aplicació. Es tracta d'un exemple de cerca d'estructures, un altre de creació de models i per últim, com realitzar el procés d'acoblament molecular. A més a més, es mostren alguns resultats obtinguts durant l'execució de l'aplicació.

Al capítol 4 es presenta el manual de l'aplicació Protein-MetReS, on s'explica pas a pas el funcionament de l'aplicació per a que l'usuari tingui una petita eina de suport. Aquest manual es trobarà en un futur al portal web del projecte MetReS.

Al capítol 5 es mostren les conclusions a les que hem arribat després de la realització del projecte i possibles millores o treball futur que posteriorment es podrà realitzar.

Al capítol 6 es presenten les diferents adreces web consultades i referències utilitzades durant el desenvolupament d'aquest projecte.

Finalment, a l'Apèndix A es mostra un exemple sobre una de les classes Java utilitzades en el projecte per tal de filtrar la informació que ens proporciona la pàgina de resultats d'un servidor web i així extreure'n només les dades útils. L'exemple mostrat és la classe ProteinDataBank.java A l'Apèndix B es mostra el codi de la classe Mail.java la qual ens permet llegir i tractar els diferents correus electrònics que es van rebent. Per últim, a l'Apèndix C es descriuen els diferents arxius de configuració del GRAMM.

Capítol 2

Aplicació Protein-MetReS

A continuació es descriu l'aplicació Protein-MetReS amb més detall, quines són les seves característiques, funcions i més descripcions que s'han cregut convenients especificar.

Com ja s'ha esmentat anteriorment, l'aplicació disposa d'una interfície gràfica a través de la qual es poden realitzar totes les operacions permeses. Primer existeix un procés d'autenticació el qual ens permet accedir a la interfície principal. Un cop s'ha accedit a la interfície, ja tenim l'opció de treballar amb els gens de l'organisme seleccionat. Amb la llista de gens seleccionats podem realitzar les diferents accions que es mencionaven anteriorment, com són la cerca d'estructures o la creació de models estructurals. Per cada opció, l'aplicació disposa de la seva interfície individual, ja que els servidors utilitzats per dur a terme cadascun dels processos són diferents. Per la cerca d'estructures disposem de servidors web mentre que per la creació de models disposem tant de servidors web com servidors locals, tot i que aquests últims no han pogut ser implementats en aquest projecte.

Un cop s'han obtingut els resultats per part dels diferents servidors, la interfície de resultats ens permet previsualitzar-los i descarregar-los. Amb els arxius de resultats obtinguts l'aplicació permet realitzar un procés d'acoblament molecular, més conegut com a docking, mitjançant una aplicació dissenyada per aquest propòsit.

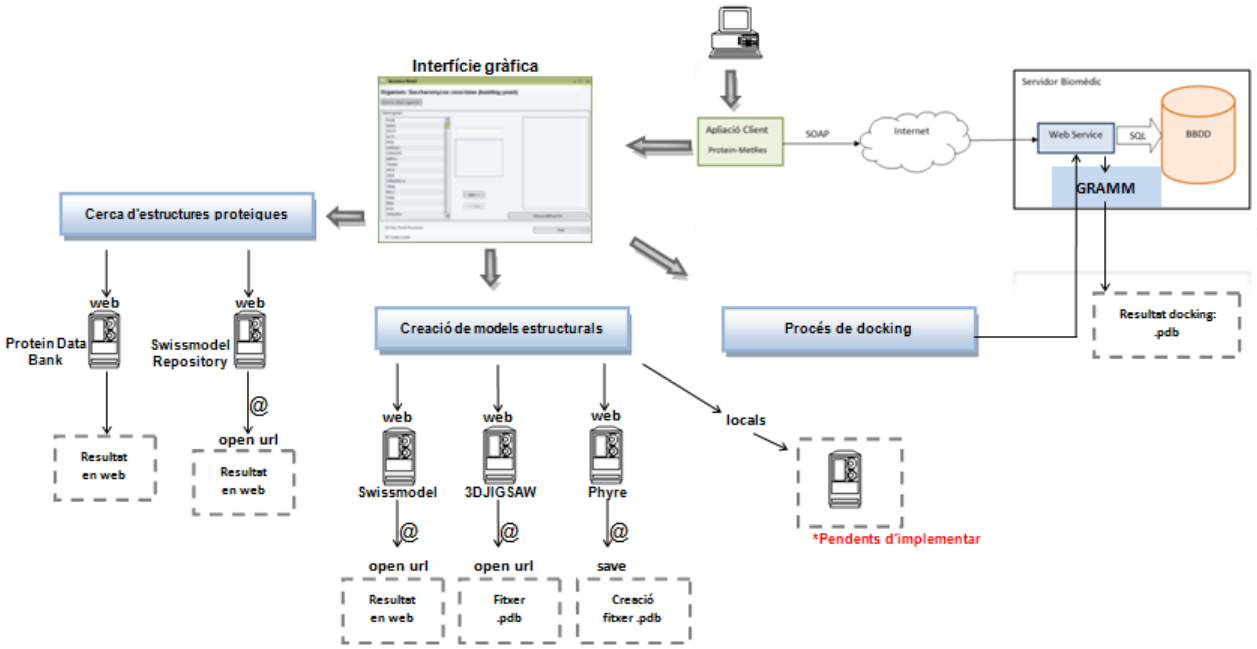


Figura 2.1: Esquema general Protein-MetReS

Ara sí, anem a veure amb més detall com es realitzen aquests processos que intervenen, amb quines dades es treballa, i com es reben i tracten els resultats, a més a més d'altres descripcions sobre aspectes que han format part en el desenvolupament de l'aplicació.

2.1 Plataforma de desenvolupament

Per al desenvolupament de l'aplicació s'ha utilitzat la plataforma Netbeans. Netbeans és una plataforma per al desenvolupament d'aplicacions d'escriptori utilitzant el llenguatge JAVA; entre altres, i un entorn de desenvolupament integrat (IDE) per desenvolupar sota aquesta plataforma. NetBeans IDE és un producte lliure i gratuït sense restriccions d'ús.

La plataforma Netbeans permet que les aplicacions siguin desenvolupades a partir d'un conjunt de components de software anomenats mòduls. Un mòdul és un arxiu Java que conté les classes de Java escrites per interactuar amb les interfícies de programació d'aplicacions (API) i un arxiu especial (manifest file[1]) que l'indica com a mòdul. Les aplicacions construïdes a partir de mòduls poden ser ampliades agregant nous mòduls.

[1]Manifest file: arxiu de contingut JAR que s'utilitza per definir les dades de l'extensió i el paquet relacionat

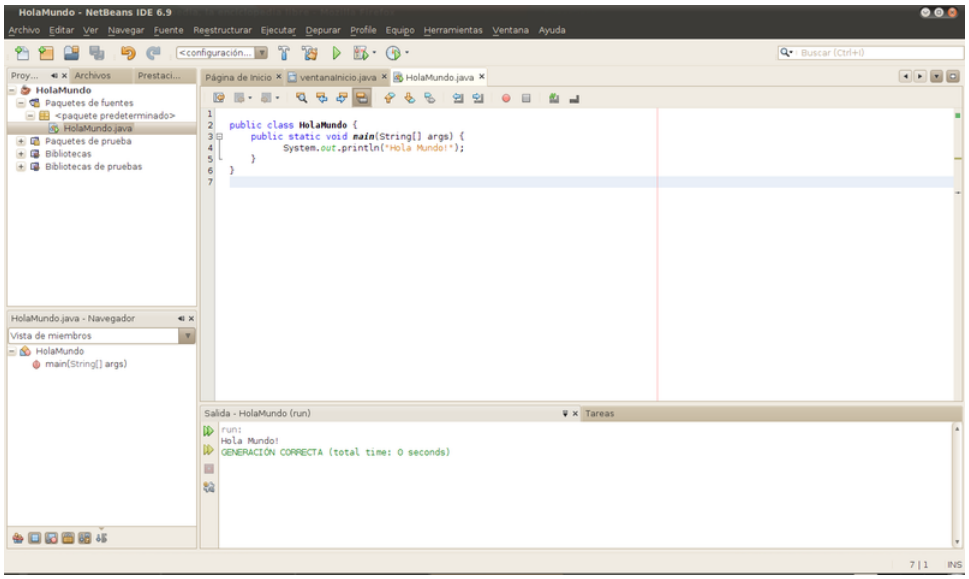


Figura 2.2: Plataforma Netbeans 6.9.1

2.2 Descripció de l'aplicació

L'aplicació Protein-MetReS és un programa que té com a funcionalitat principal tant la creació de models estructurals com la cerca d'estructures moleculars. A més a més, l'aplicació ens dóna la possibilitat de realitzar el procediment d'acoblament molecular, més conegut com docking de proteïnes. La descripció d'aquest procés es pot veure amb més detall més endavant.

Per permetre la interacció, l'aplicació Protein-MetReS utilitza una interfície gràfica d'usuari (GUI). A continuació s'explica el flux de funcionament d'aquesta aplicació:

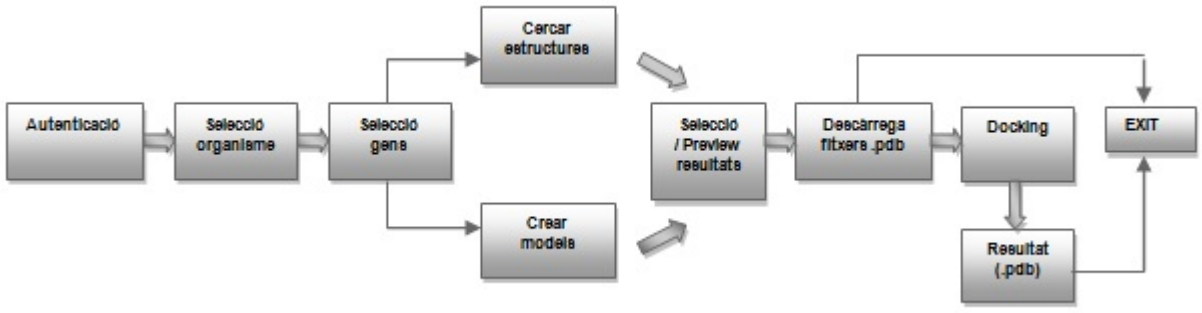


Figura 2.3: Flux d'execució

Primer de tot, per poder utilitzar l'aplicació s'ha d'iniciar sessió autentificant l'usuari que s'ha donat d'alta a la web del projecte. Aquest procés d'autenticació consisteix en comparar les dades introduïdes amb les dades d'usuaris registrats a la base de dades del servidor biomèdic.

Un cop iniciada la sessió es mostra una finestra de selecció d'un organisme a estudiar. Aquesta finestra ens dona la possibilitat de seleccionar un organisme a analitzar i d'escollir si volem seleccionar una llista predefinida de gens o no. Si escollin que si, en la finestra que ens apareix podem inserir el nom dels diferents gens en els quals estem interessats. Pel contrari, si decidim carregar tots els gens de l'organisme, en la següent finestra; en aquest cas la finestra principal, podem veure la llista de gens que podem seleccionar. Una vegada hem seleccionat l'opció desitjada només ens falta prémer el botó "Start". Segons l'opció escollida anteriorment, s'ens mostrarà una finestra o una altra, on haurem d'escollir quins servidors volem que processin la informació. Posteriorment, un cop obtinguts els resultats, una nova interfície ens dóna la possibilitat d'escollir el fitxer ".pdb" desitjat per cada gen. Fins i tot, es té la possibilitat de poder previsualitzar el fitxer abans d'escollir-lo. Un cop s'han descarregat les fitxers ".pdb" l'usuari té l'opció de realitzar el procés de docking amb els fitxers que està treballant en aquell moment o de sortir de l'aplicació. En el cas d'haver escollit la primera opció, la interfície de docking ens permet escollir les parelles de fitxers desitjades per al procediment de l'acoblament molecular i, posteriorment, seleccionar la configuració dels paràmetres que intervindran al procés de docking. Per últim, l'usuari rep els resultats del procediment per posteriorment ser tractats.

S'ha pogut conèixer breument el flux d'execució de l'aplicació, però a continuació es detallaran les opcions més importants de l'aplicació, com són crear models, cercar estructures o el procés de docking. A més a més, la descripció de la interfície gràfica i el procediment detallat per fer ús de l'aplicació Protein-MetReS es presenta al manual d'usuari que es troba al Capítol 4.

2.2.1 Crida a aplicacions externes

Durant el disseny de l'aplicació Protein-MetReS i fent ús de les possibilitats que ens ofereix el llenguatge JAVA, s'ha hagut d'executar algunes aplicacions externes o comandes. Cal dir que aquestes aplicacions externes o comandes s'han executat des del web service. Per realitzar aquest tipus de crides, s'ha fet ús de la classe Runtime, la qual ens permet executar comandes del sistema operatiu:

- Tan sols existeix una instància de classe Runtime, que s'obté amb:

  Runtime rt = Runtime.getRuntime();

- L'objecte obtingut amb la crida Runtime.getRuntime() ens permet executar processos amb:

  rt.exec ("ruta del programa / programa");

```
public class externalProgram {
  public static void main(String[] args){
  try
    {
      Runtime rt = Runtime.getRuntime();
      Process n = rt.exec("/usr/bin/mi_programa");
      n.waitFor();
    }
    catch ( IOException ioe )
    {
        ioe.printStackTrace();
    }
    catch ( InterruptedException ie )
    {
        ie.printStackTrace();
    }
  }
}
```

Figura 2.4: Exemple crida aplicació externa

2.2.2 Web Service

Una vegada el servidor biomèdic ha estat preparat per poder oferir suport a totes les funcionalitats que es necessiten desenvolupar, s'ha passat a la fase de distribució de l'aplicació.

Per dur a terme aquesta distribució, cal especificar que l'aplicació Protein-MetRes utilitza una base de dades per poder funcionar, la qual està integrada en el servidor i l'accés a la mateixa està restringit a usuaris i aplicacions autoritzats.

Per aquest motiu, la decisió va ser que l'aplicació es transformés en una aplicació-client d'un servei web ofert pel servidor. Aquest servei web s'ha estès sobre Tomcat, el qual també és considerat un servidor web, ja que incorpora el seu propi servidor HTTP.

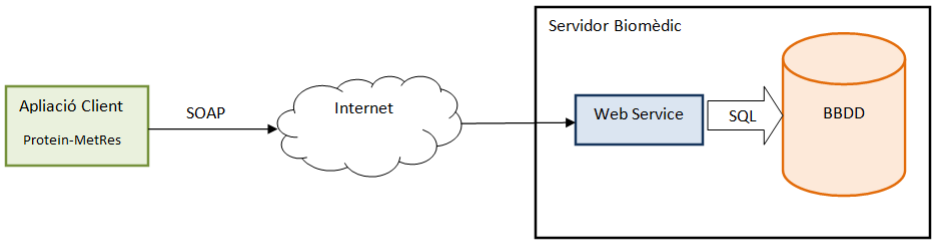


Figura 2.5: Model esquemàtic Web Service

Aquest web service s'ha desenvolupat utilitzant la tecnologia JAX-WS (Java and XML Web Service) de Java, la qual integra tecnologia SOAP (Simple Object Access Protocol) per a la comunicació entre l'aplicació client i el servei web. El desenvolupament del web service ha de tenir un seguit de funcionalitats, ja sigui l'autenticació d'usuaris o permetre a l'aplicació client accedir a la base de dades.

Com s'ha comentat anteriorment, el David Terés va implementar el WebService oferint una serie de serveis per a l'aplicació Biblio-MetReS. Degut a les necessitats de la nostra aplicació, s'ha afegit tres serveis web més. Els serveis web es defineixen de la següent manera:

```
@WebMethod(name)
@RolesAllowed
public <type>
function(values){
desenvolupament.
}
```

- L'etiqueta "@WebMethod" conté l'identificador de la funcionalitat. Aquest identificador l'utilitzen les aplicacions client per fer la crida al servei ofertat.
- L'etiqueta "@Roles Allowed" identifica quins rols d'usuari poden fer ús d'aquesta funcionalitat. Cada usuari tindrà un o més rols, aquests rols estan representats a la base de dades i mapejats en un dels fitxers XML del Web Service.
- El contingut del WebMethod és idèntic a una funció estàndard de Java, l'única peculiaritat és que està obligat a retornar un valor a l'aplicació client, ja sigui un resultat o una excepció d'error.

Els serveis web implementats són:

- ***uploadPDB(name, mail)***: aquesta funció puja al servidor els fitxers pdb necessaris per a realitzar el docking. En cada crida es puja un fitxer. A més a més, crea un directori temporal per poder realitzar tot el procés de docking. Aquest directori rep el nom del e-mail de l'usuari.
- ***executeGRAMM(mail, pairs)***: aquesta funció prepara els fitxers de configuració del Gramm i executa la comanda *gramm scan coord* per realitzar el docking en el directori creat en la funció anterior.
- ***downloadPDBs(mail, pair)***: aquesta funció descarrega els fitxers pdb resultants del docking. En cada crida es descarrega un fitxer.

Per fer la crida al WebService s'utilitza la següent sintaxis:

source.WSMetRESService service = new source.WSMetRESService();

source.WSMetRES port = service.getWSMetRESPort();

Map requestContext = ((BindingProvider) port).getRequestContext();

requestContext.put(BindingProvider.USERNAME_ PROPERTY, login);

requestContext.put(BindingProvider.PASSWORD_ PROPERTY, pass);

2.2.3 Detecció de camps

Abans de detallar els processos de cerca d'estructures i creació de models, explicarem quin mètode hem seguit a l'hora de detectar els camps necessaris per tal de dur a terme les peticions corresponents de forma correcta. Una de les opcions podria ser introduir els camps directament en la barra d'adreces en el navegador, però cal dir que aquest no és el cas. En la nostra aplicació hem hagut de detectar quins camps eren necessaris per realitzar la petició en cada servidor i posteriorment tractar-los en el nostre codi. Per realitzar aquesta detecció hem fet ús de l'extensió *Firebug* del navegador Firefox. Aquesta extensió ens permet veure el funcionament intern dels llocs webs, la qual cosa ens permet analitzar el codi amb més detall. A continuació tenim un exemple:

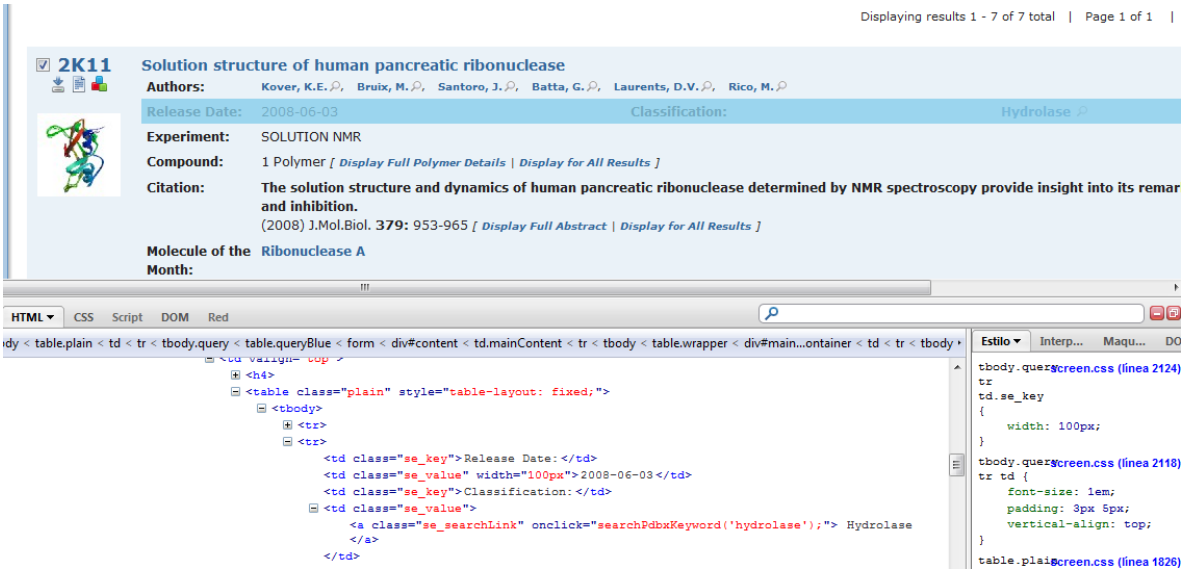


Figura 2.6: Detecció de camps mitjançant Firebug

I seguint aquest procés, s'han realitzat totes les deteccions de camps de tots els llocs webs on s'allotgen els diferents servidors als quals s'han realitzat les peticions.

2.2.4 Servidors: funcionament

En aquest apartat es pot observar quins són els servidors que actuen dins de l'aplicació, ja sigui per la cerca d'estructures com per la creació de models.

Per fer això es necessita enviar la seqüència d'aquest gen, que com ja s'ha explicat, és una cadena de lletres que conté l'estructura primària d'una molècula. Aquesta seqüència s'adquireix interactuant amb la base de dades del servidor biomèdic quan es selecciona un gen, i un cop obtinguda s'envia juntament amb la consulta a cada servidor seleccionat.

Un cop enviada la petició, aquesta seqüència es comparada amb les bases de dades de cada servidor per obtenir l'estructura que la defineix. En cas de no trobar-la el programa et dóna la possibilitat de crear els models per homologia, mitjançant els servidors de creació del model. En aquest cas el resultat obtingut és una predicció sobre l'estructura que la conformarà.

A continuació s'explica el funcionament de cada servidor, un esquema de com es fa cada petició i els paràmetres necessaris per fer-la.

2.2.4.1 Cerca d'estructures

Com s'ha comentat anteriorment, una vegada estem en la pantalla principal de l'aplicació tenim la possibilitat de cercar estructures una vegada hem escollit de la llista els gens que desitgem tractar. En què consisteix la cerca d'estructures? A continuació es detalla el procés.

Un cop apareix la finestra de cerca d'estructures es poden seleccionar dos servidors. La petició es pot fer tan individualment com a ambdós a la vegada.

Protein Data Bank: l'aplicació genera la petició al servidor (portal web) passant-hi els camps necessaris per fer la consulta. Una vegada el servidor retorna la pàgina web de resultats, aquests es tracten per posteriorment ser mostrats correctament a la interfície de resultats. Per tractar aquests resultats s'ha fet ús del paquet Java javax.swing.text.html.*; entre altres. Aquest paquet ens ha permès poder analitzar el codi HTML i filtrar tota la informació desitjada fent ús de les etiquetes HTML. Per filtrar aquesta informació s'ha treballat principalment amb tres funcions que són les següents:

- public void handleEndTag(HTML.Tag tag, int pos){} → implementació del codi per detectar el final del tag HTML.
- public void handleStartTag(HTML.Tag tag, MutableAttributeSet a, int pos){} → implementació del codi per detectar l'inici del tag HTML.
- public void handleText(char[] data, int pos){} → implementació del codi per extreure el text entre les etiquetes i/o tags.

A l'Apèndix A es pot veure amb detall la classe utilitzada per filtrar la informació.

Aquest servidor té el següent esquema d'execució:

Figura 2.7: Esquema d'execució servidor Protein Data Bank

I els paràmetres necessaris per fer la petició són els següents:

Enumeration en=sequence.keys();
String key=en.nextElement().toString();
String s=(String) sequence.get(key);
String web="http://www.pdb.org/pdb/search/navbarsearch.do?inputQuickSearch=";
String url=web+s;
ProteinDataBank pdb=new ProteinDataBank(logger, url);
listPDB=pdb.Execute();

SwissModel Repository: a l'igual que en el servidor anterior l'aplicació genera la petició mitjançant el mateix procés anterior però el tractament de la informació és diferent. En aquest cas, el servidor retorna el resultat per correu. Arriba un e-mail a la direcció de correu especificada a l'inici de la petició i l'aplicació Protein-MetReS s'encarrega de detectar un nou e-mail en la safata d'entrada. Un cop el detecta, el llegeix i obté el link que s'encarrega de mostrar-nos els resultats. Per últim, una vegada obert el link de la pàgina de resultats, l'aplicació actua de la mateixa manera que en el servidor anterior. Fa ús de les funcions anteriors per poder filtrar la informació de la pàgina de resultats i mostra els resultats d'igual forma.

Un cop finalitzada la petició es mostra la interfície de resultats on l'usuari pot visualitzar la informació de tots els resultats trobats de forma clara i organitzada i/o previsualitzar el fitxer que retorna aquesta execució abans de procedir a la descàrrega d'aquest fitxer. L'esquema d'execució és el següent:

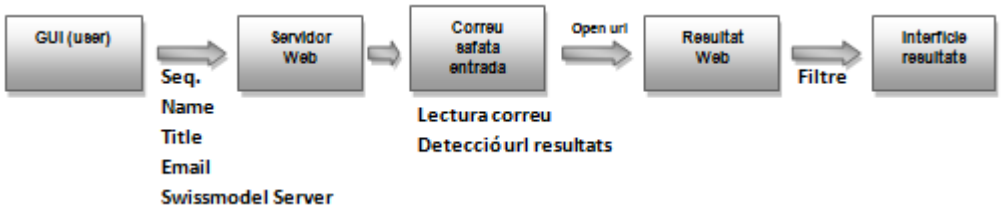


Figura 2.8: Esquema d'execució servidor SwissModel repository

Els paràmetres que necessita per realitzar la petició adequadament són els següents:

String web= "http://www.proteinmodelportal.org/?aid=queryModellingInteractiveBySeq&zid=async";
String seq="&seq="+s;
String name="&tools_name=MetReS";
String title="&tools_title=Biblio-MetReS";
String email="&tools_email=metres.cmb@gmail.com";
String swissmodelServer="&tools_smw=1";

URL swissmodel = new URL(web+name+title+email+seq+swissmodelServer);
swissmodel.openStream();

2.2.4.2 Creació de models estructurals

Un cop situats en la pantalla principal, si l'usuari el que desitja és crear un model amb cadascun dels gens seleccionats ho pot fer escollint un o diversos servidors que destaquem a continuació.

- Servidors Web. Més avall s'explica el funcionament de cadascun d'ells de forma detallada.
  - SwissModel.
  - 3DJIGSAW.
  - PHYRE.
- Servidors locals (No implementats).
  - Rosetta: És un servidor que proporciona eines automatitzades per la predicció d'estructures de proteïna i el seu anàlisi. L'adreça web del servidor és: http://robetta.bakerlab.org/submit.jsp
  - Modeller: S'utilitza per a la homologia o modelatge comparatiu d'estructures tridimensionals de proteïnes. L'adreça web del servidor és: https://modbase.compbio.ucsf.edu/scgi/modweb.cgi

– MetaServer: Ofereix una porta d'entrada a l'estructura de proteïnes fent una comparativa entre els mètodes de predicció de la funció. L'adreça web del servidor és:
http://meta.bioinfo.pl/submit_wizard.pl

En aquest cas, els servidors web actuen tots d'igual forma, és a dir, tots envien un e-mail a la safata d'entrada de la direcció @ especificada. Però bé, anem a explicar quin procés segueixen cadascun d'ells.

SwissModel: Una vegada feta la petició, l'aplicació espera a que arribi un e-mail en la carpeta corresponent (s'han creat carpetes i filtres en en compte de correu). Una vegada l'ha detectat el llegeix i obté el link de resultats que posteriorment són tractats per ser mostrats en la pantalla de resultats.

L'esquema d'execució d'aquest servidor és el següent:

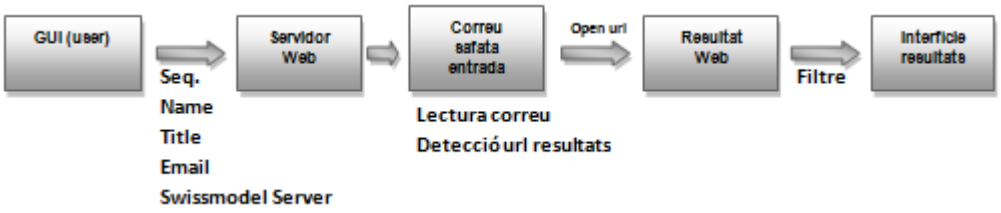


Figura 2.9: Esquema d'execució servidor SwissModel

Els paràmetres que necessita són:

```
String web = "http://www.proteinmodelportal.org/?aid=queryModellingInteractiveBySeq&zid=async";
String name = "&tools_name=MetReS";
String title = "&tools_title="+gene+"-"+login;
String email = "&tools_email=metres.cmb@gmail.com";
String swissmodelServer = "&tools_smw=1";
String seq="&seq="+seqüencia_proteina;

URL swissmodel = new URL(web + name + title + email + seq+ swissmodelServer);
swissmodel.openStream();
```

3DJIGSAW: Després de fer la petició, l'aplicació espera a que arribi un e-mail en la seva carpeta a l'igual que el servidor anterior, amb la diferència que en aquest cas el e-mail de resultats incorpora un fitxer adjunt, el qual es correspon amb el fitxer amb extensió ".pdb". Més endavant podrem observar de que es tracta aquest tipus de fitxer. Per tant, en aquest cas l'aplicació detecta el link de baixada i descarrega el fitxer. Posteriorment, l'usuari podrà visualitzar i/o interpretar els resultats.

L'esquema d'execució d'aquest servidor és el següent:

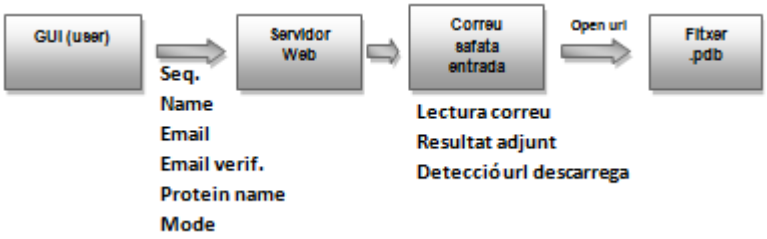


Figura 2.10: Esquema d'execució servidor 3DJIGSAW

Els paràmetres que necessita són:

```
String web = "http://bmm.cancerresearchuk.org/~3djigsaw/scripts/3djigsaw_submit_maestro.cgi?";
String realname = "realname=Biblio-MetReS";
String email = "&email=metres.cmb@gmail.com";
String email2 = "&email2=metres.cmb@gmail.com";
String protein_name = "&Protein%20name="+gene;
String mode = "&mode=automatic";
String seq="&Sequence="+seqüencia_proteina;

URL djigsaw = new URL(web + realname + email + email2 + protein_name + mode + seq);
djigsaw.openStream();
```

PHYRE: En aquest cas, el servidor actua de forma diferent als anteriors. Una vegada realitzada la petició i rebut el correu, ens trobem amb que el correu no conté cap tipus d'enllaç ni fitxer adjunt on es mostrin els resultats. És el propi correu en sí qui conté els resultats. Per tant, l'aplicació llegeix el correu byte per byte al mateix temps que el guarda com un fitxer dins de la nostra computadora.

L'esquema d'execució d'aquest servidor és el següent:

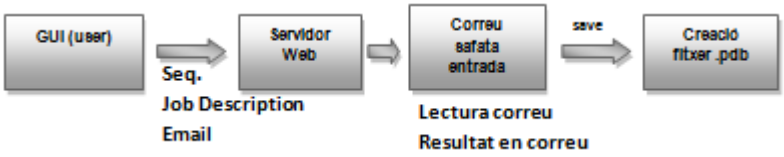


Figura 2.11: Esquema d'execució servidor Phyre

Els paràmetres que necessita són:

```
String web = "http://www.sbg.bio.ic.ac.uk/phyre/qphyre_scripts/qphyre_submit.cgi?";
String jobDescrip = "&seq-desc="+gene;
String email = "&usr-email=metres.cmb@gmail.com";
String seq="&sequence="+seqüencia_proteina;

URL phyre = new URL(web + email + jobDescrip + seq);
phyre.openStream();
```

Els servidors retornen el resultat mitjançant un correu electrònic. Anteriorment, quan parlàvem dels servidors web que hem utilitzat a l'hora de crear models, hem parlat de la creació de carpetes i filtres en el compte de correu on volíem rebre els resultats de les peticions. A continuació us mostrem els dos passos que s'han seguit per configurar el nostre compte de correu, en aquest cas de Gmail.

1. Crear etiqueta nova des de el menú Administrar Etiquetes o directament Crear Etiqueta Nueva

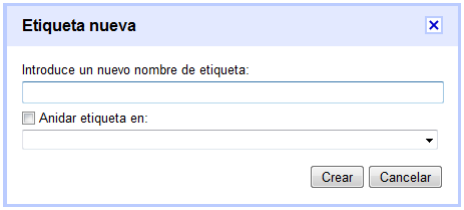


Figura 2.12: Etiqueta/carpeta nova

2. Des de el menú Administrar Etiquetes – Filtres podem crear tants filtres com necessitem, en el nostre cas un per cada adreça de correu que ens retornava cadascun dels servidors que així actuaven.

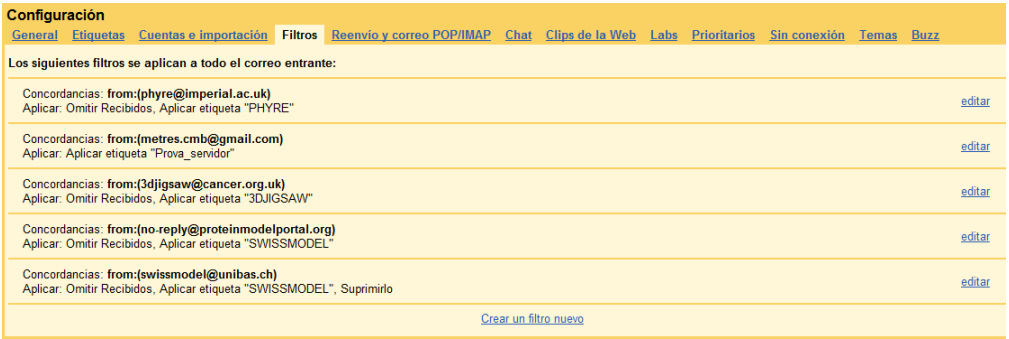


Figura 2.13: Configuració de filtres

2.2.5 Mail

Per obtenir la sessió de correu i accedir a les diferents carpetes del compte, hem hagut d'utilitzar l'expansió de Java anomenada JavaMail, a més a més per ser funcional necessita una llibreria addicional, JAF (JavaBeans Activation Framework). A continuació anem a veure un petit fragment de codi on podrem observar com s'obté la sessió del correu i com s'obté la carpeta per llegir els correus. En l'Apèndix B es troba amb detall la classe Mail utilitzada en l'aplicació Protein-MetReS.

```
//We have to obtain the session mail
Properties prop = new Properties();
prop.setProperty("mail.imap.starttls.enable","false");
prop.setProperty("mail.imap.socketFactory.class","javax.net.ssl.SSLSocketFactory");
prop.setProperty("mail.imap.socketFactory.fallback", "false");
prop.setProperty("mail.imap.port", "993");
prop.setProperty("mail.imap.socketFactory.port", "993");
Session sesion = Session.getInstance(prop);
//We obtain the Store and the Folder to read mails.
Store store = sesion.getStore("imap");
store.connect("imap.gmail.com", "user@gmail.com", "user1234");
Folder folder = store.getFolder(directori);
folder.open(Folder.READ_ WRITE);
```

2.2.6 Estructura PDB

Anteriorment s'ha esmentat el fitxer .pdb. A continuació s'explica en què consisteix aquest tipus de fitxer i quina estructura presenta.

El fitxer .pdb és un arxiu de coordenades moleculars que conté informació precisa sobre l'estructura tridimensional d'una molècula. És un fitxer d'accés públic i pot aconseguir-se en diferents llocs a través del WEB, com per exemple, un dels mes coneguts i comentats anteriorment Protein Data Bank.

En ell es troben els arxius de coordenades moleculars per a la majoria de molècules on la seva estructura terciària ha estat clarificada per les tècniques de difracció de RAIGS X i Ressonància Magnètica Nuclear, com també els arxius de coordenades construïts mitjançant l'aplicació de models matemàtics.

Per poder interpretar un fitxer PDB s'hauria de saber interpretar cadascuna de les línies/registres o capçaleres que conté, però bé, aquest no és el propòsit d'aquest projecte tot i que a continuació podem veure de forma resumida les línies/registres que el composen i la seva descripció.

HEADER	*Nom i data de creació de l'arxiu PDB*
COMPND	*Nom de la molècula*
SOURCE	*Organisme del que es va obtenir la proteïna*
AUTHOR	*Llista dels autors que van proporcionar l'estructura molecular al PDB*
REVDAT	*Dades de revisió de l'estructura de la proteïna*
REMARK	*Comentaris en relació als articles on es va publicar l'estructura molecular o sobre les característiques de la molècula*
SPRSDE	*Llista d'arxius de coordenades per a la mateixa estructura*
SEQRES	*Seqüència dels aminoàcids de la proteïna*
FTNOTE	*Notes de peu de pàgina. No tots els arxius ho tenen*
HET & FORMUL	*Llista de cofactors, grups prostètics, inhibidors i altres substàncies no proteiques presents*
HELIX, SHEET & TURN	*Llista dels residus amb estructura secundaria en la proteïna*
CRYST1, ORIG & SCALE	*Informació general sobre els cristalls amb els quals es va obtenir l'estructura per RAIG X*
ATOM & HETATM	*Conté la informació de la posició espacial en els eixos X,I i Z de cadascun dels àtoms, especificant el residu i la cadena*
CONECT	*Enllaços formats entre els àtoms no proteics presents en el PDB*
MASTER & END	*Indiquen la finalització de l'arxiu*

Taula 2.1: Descripció de les línies/registres d'un fitxer PDB

Les línies ATOM, tenen l'estructura següent:

ATOM 29 CA PRO A 5 -2.551 -37.351 4.539 1.00 0.00 1CLG 79

- El nombre seqüencial de l'àtom dins de l'arxiu. (**29**)
- El tipus d'àtom (**CA**):
  - N = Nitrogen del grup Amida
  - CA = Carboni alpha
  - O = Oxigen del grup Carbonil
  - CB = Carboni beta
  - CD= Carboni delta
- Nombre del residu, codi de tres lletres (**PRO**) .
- Nombre del residu i cadena (**A-5**).
- Coordenades en angstroms de l'àtom en els eixos X, I i Z de la cel·la unitària. (**-2.551 -37.351 4.539**).

- Ocupància : Fracció de la cel·la unitària que conté l'àtom en una ubicació particular. Normalment aquest valor és d'**1.00** .
- Factor de temperatura : és una indicatiu de la incertesa de la posició de l'àtom a causa del desordre per les vibracions tèrmiques. (**0.00**).
- El codi d'identificació ID de l'arxiu PDB. (**1CLG**).

Aquesta informació pot llegir-se amb un editor de text normal com Word o el Bloc de Notes. És important tenir en compte que en canviar algun d'aquests paràmetres s'afecta l'estructura molecular en conjunt. Per això, de ser necessari, els canvis realitzats han de guardar-se amb un nom diferent al codi d'identificació ID assignat pel PDB.

2.2.7 Procés de Docking

El docking o acoblament molecular es pot considerar com l'optimització d'un mecanisme, el qual es podria descriure com el millor encaix del ligand que s'uneix a un receptor. No obstant això, tant el ligand com el receptor són flexibles. Durant el procés, tots dos ajusten la conformació per aconseguir el millor acoblament global. L'objectiu és aconseguir una conformació òptima tant per al receptor, com per al seu ligand, fent que l'orientació entre ambdós minimitzi l'energia lliure.

En termes generals, és un mètode que s'empra per predir la conformació preferida d'una molècula, quan està unida a una altra, amb la finalitat de formar un complex estable. Es pot pensar amb el procés de docking com un objecte "clau-tancament", on el més interessant és conèixer l'orientació relativa de la "clau", la qual obrirà el "tancament". El receptor es pot veure com el "tancament" i el ligand (molècula que s'unirà a la proteïna) com la "clau".

Per fer aquest procés s'agafa un ligand i es va traslladant i rotant per tot el centre actiu, generant diferents orientacions. Cadascuna d'aquestes orientacions avalua l'energia de la unió del complex que es forma. Quan menor és l'energia, millor és la interacció, i per tant més estable serà el complex que es forma.

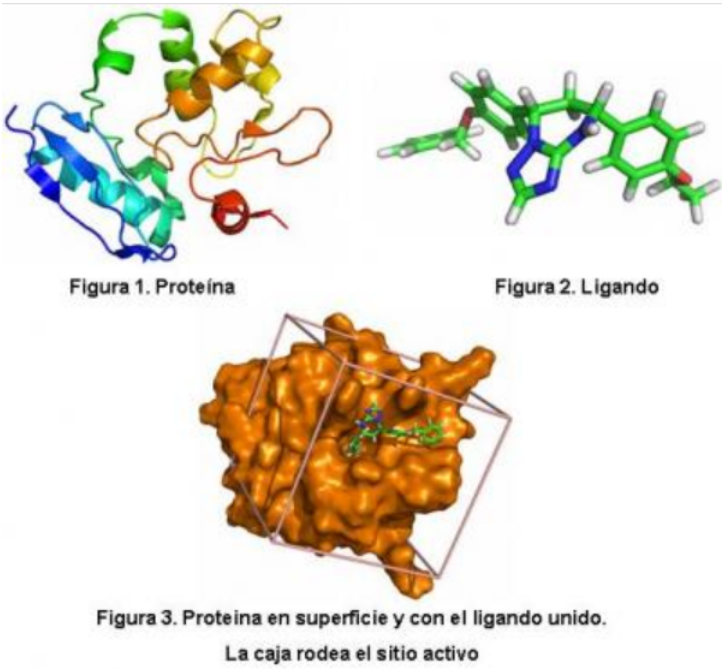


Figura 2.14: Exemple procés de docking

En el nostre cas, i per la realització d'aquest procés, hem utilitzat l'aplicació Global RAnge Molecular Matching (GRAMM). És un programa per l'acoblament de proteïnes que per predir l'estructura del complex, solament necessita les coordenades atòmiques de les dues molècules. El programa realitza una cerca exhaustiva en 6-D a través de les traduccions i rotacions de les molècules.

El programa presenta algunes configuracions que l'usuari hauria de tenir presents. L'aplicació disposa de tres fitxers de configuració que es presenten a continuació:

- rpar.gr (parameters)
- rmol.gr (molecules description)
- wlist.gr (list of results)

Cada fitxer conté la configuració dels diferents paràmetres per dur a terme el procés de docking. En aquest apartat no entrarem en el detall d'aquests, ja que en l'Apèndix C s'adjunta el format d'aquests fitxers i el significat dels seus respectius paràmetres.

A continuació es pot observar el procés d'execució de l'aplicació.

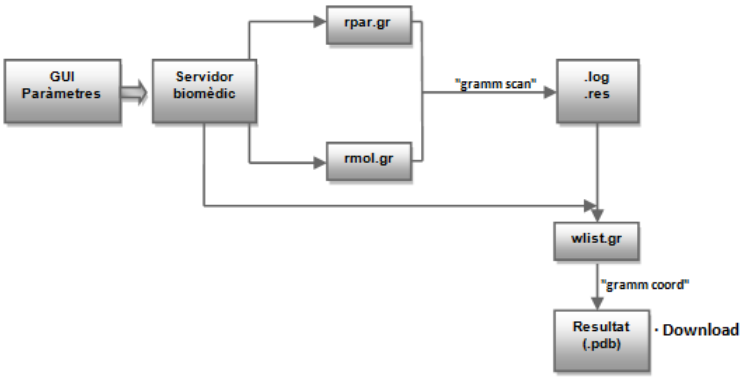


Figura 2.15: Execució procés docking

Quan l'usuari decideix realitzar el procés de docking, una vegada ha seleccionat les diferents parelles amb les quals es realitzarà el procés, disposa d'una interfície on pot configurar els paràmetres que intervindran en el docking. Aquests paràmetres s'envien al servidor biomèdic que s'encarregarà d'executar les comandes i crides pertinents i generarà els diferents fitxers que necessita el GRAMM per dur a terme el procés. Amb la comanda "gramm scan" es generen el fitxers .log i .res (resultats). Una vegada es disposa del fitxers de resultats s'ha de crear un nou fitxer, en aquest cas l'arxiu wlist.gr, i amb la comanda "gramm coord" l'aplicació genera el resultat final.

Una vegada el programa retorna els resultats, aquests poden ser visualitzats i analitzats amb una eina anomenada Swiss-PdbViewer. Aquesta aplicació proporciona una interfície fàcil d'utilitzar la qual permet analitzar varies proteïnes al mateix temps.

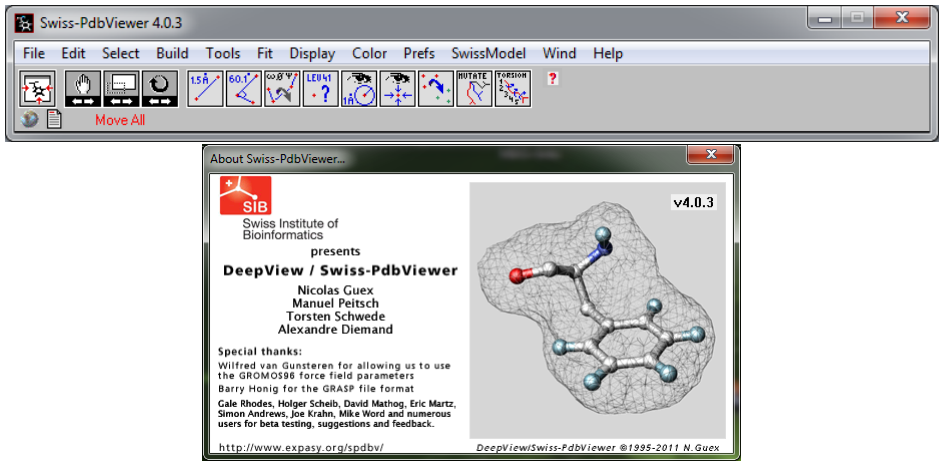


Figura 2.16: Swiss-PdbViewer 4.0.3

2.3 Diagrama de classes

En aquest apartat es presenta el següent diagrama amb els atributs, relacions d'herència i associacions de les classes utilitzades en aquest projecte.

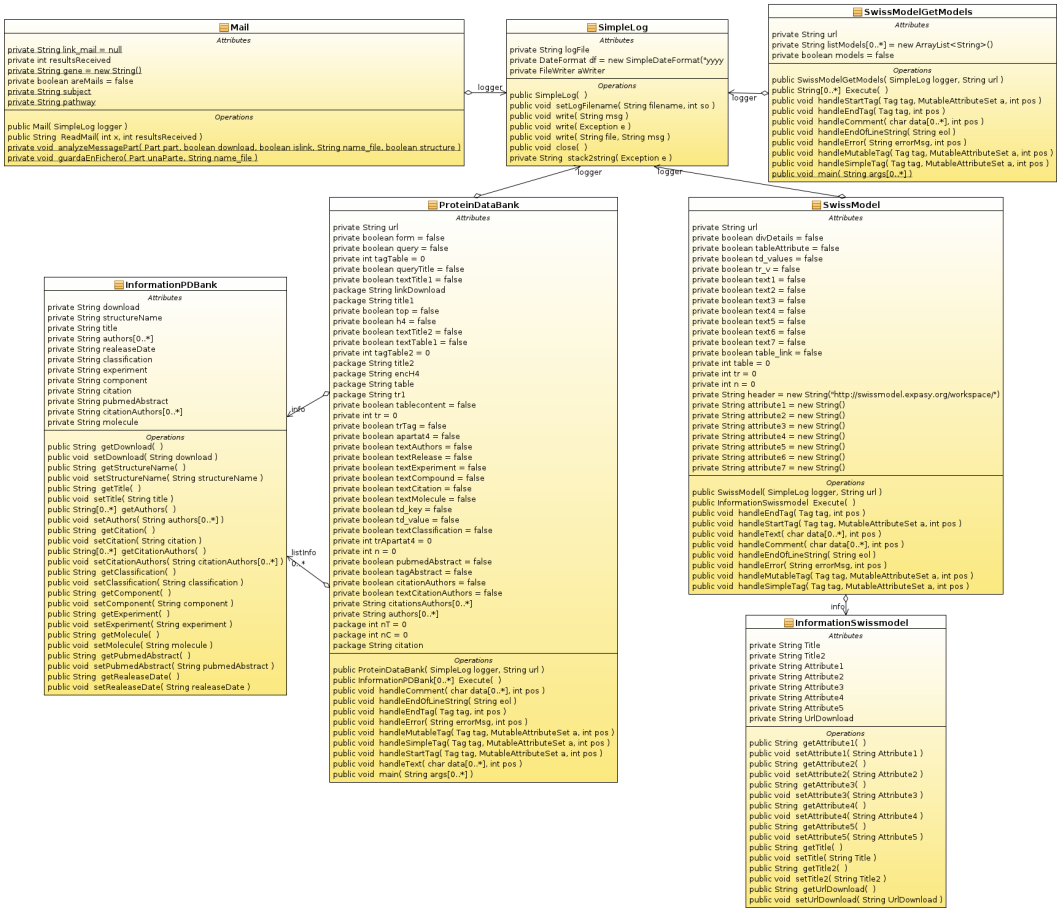


Figura 2.17: Diagrama de classes

Com es pot observar en el diagrama de classes, existeixen diferents classes que actuen com a classes principals. A continuació es pot veure una petita descripció de les operacions que realitzen.

- *Mail.java*: s'encarrega de llegir els correus que van arribant als diferents directoris de la safata d'entrada i en tot cas, d'obtenir el fitxer .pdb.
- *InformationPDBank.java*: permet l'emmagatzemament de les dades filtrades per la classe *ProteinDataBank.java*.

- *ProteinDataBank.java*: s'utilitza per obtenir les dades útils de la pàgina de resultats que ens proporciona el servidor web.
- *SwissModel.java*: a l'igual que la classe anterior, s'utilitza per obtenir les dades útils de la pàgina de resultats que ens proporciona el servidor web.
- *SwissModelMail.java*: s'utilitza per extreure el link del model de dins de la pàgina de resultats a diferencia de la classe anterior que extreu tota la informació referent a l'estructura.
- *InformationSwissmodel.java*: permet l'emmagatzemament de les dades filtrades per la classe *Swissmodel.java*
- *SwissModelGetModels.java*: en el cas que la pàgina de resultats del servidor contingui més d'un model com a resultat, s'utilitza per obtenir l'enllaç de cadascun d'ells.

La classe *ProteinDataBank.java* permet obtenir de la URL de resultats que ens proporciona el servidor web Protein Data Bank, totes aquelles dades útils amb les quals treballarem, és a dir, actua com un filtre. Amb la classe *InformationPDBank.java* emmagatzemem aquestes dades obtingudes.

La classe *SwissModel.java* permet obtenir la URL de resultats que ens proporciona el servidor web SwissModel, totes aquelles dades útils amb les quals treballarem és a dir, actua com a filtre. A més a més, en el cas que el servidor retorni més d'un model com a resultat, la classe *SwissModelGetModels.java* s'encarrega d'obtenir la URL de cadascun d'ells. Amb la classe *InformationSwissmodel.java* emmagatzemem aquestes dades obtingudes.

Capítol 3

Cas Pràctic

A continuació es mostra un exemple d'execució d'aquesta aplicació.

El primer pas és accedir amb un nom d'usuari i contrasenya, els quals es poden obtenir fent un registre a la pàgina web http://metres.udl.cat/registre.

Figura 3.1: Pas 1: Login

3.1 Selecció d'organismes

Un cop hem accedit al programa el següent pas és escollir l'organisme sobre el qual es treballa, en aquest cas el Saccharomyces cerevisiae (budding yeast), com és mostra a la figura 3.2.

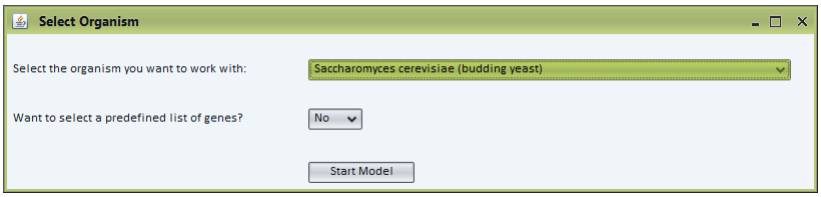


Figura 3.2: Pas 2: Escollir organisme

A continuació es seleccionen els Gens, sobre els quals tenim dues opcions:

- Cercar Estructures
- Crear Models

3.1.1 Cercar Estructures

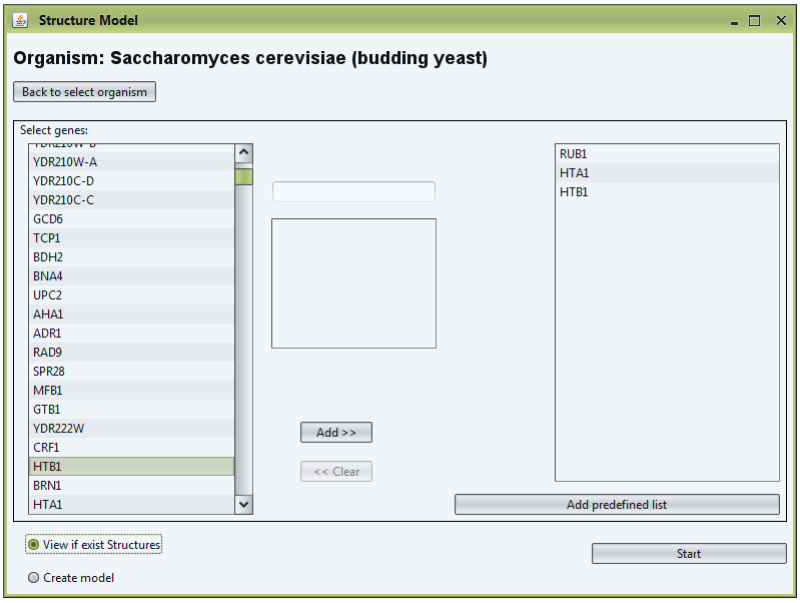


Figura 3.3: Selecció de gens

S'escolleixen els Gens dels que es vol cercar les estructures i a continuació, els servidors sobre els que es fa la cerca. Figura 3.4.

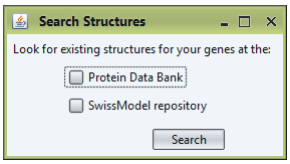


Figura 3.4: Cercar Estructures

Un cop fet això s'obtenen els resultats, que es mostren en una taula on es pot visualitzar tota la informació sobre cada gen trobat i/o visualitzar el fitxer "*.pdb"que se n'obté de la cerca abans de descarregar-los.

3.1.2 Crear Models

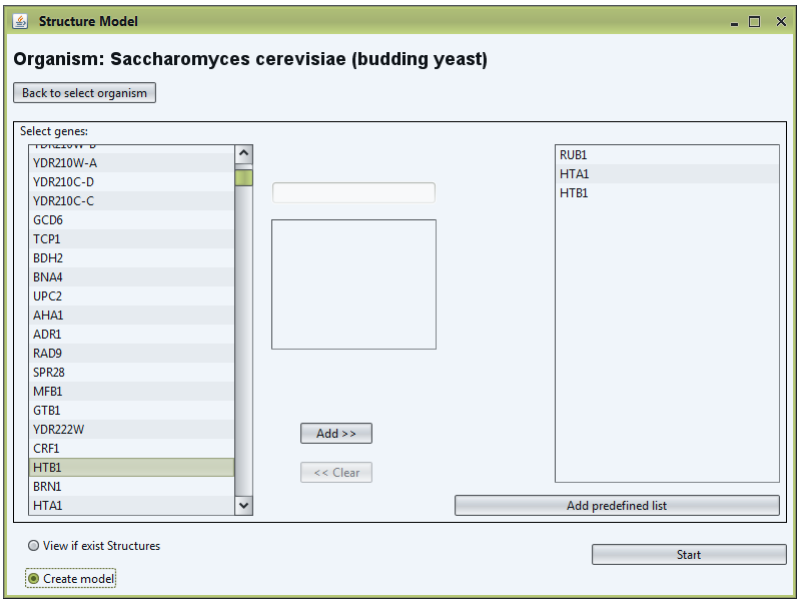


Figura 3.5: Crear Models

Un cop escollits els gens es procedeix a la creació de models escollint els servidors web que es volen utilitzar, SwissModel, 3DIJGSAW i/o Phyre.

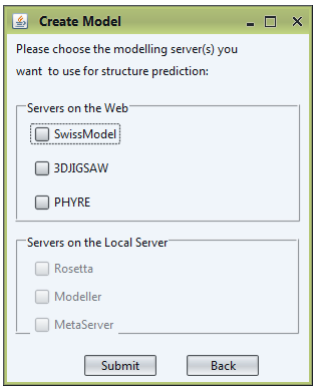


Figura 3.6: Servidors de creació de models

3.2 Tractament dels resultats

Un cop obtinguts els resultats tant en la cerca d'estructures com en la creació del models, aquests, es mostren en una taula on inclou el nom del gen, el nombre de fitxer ".pdb" trobats i on es pot escollir el fitxer a descarregar per cada gen.

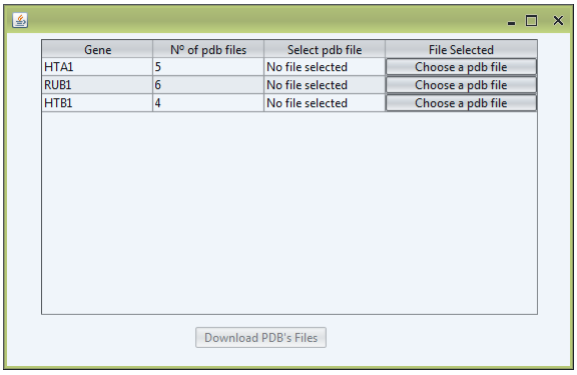


Figura 3.7: Resultats trobats

Però abans de descarregar els fitxers l'usuari té la oportunitat de veure tota la informació sobre els fitxers a descarregar,

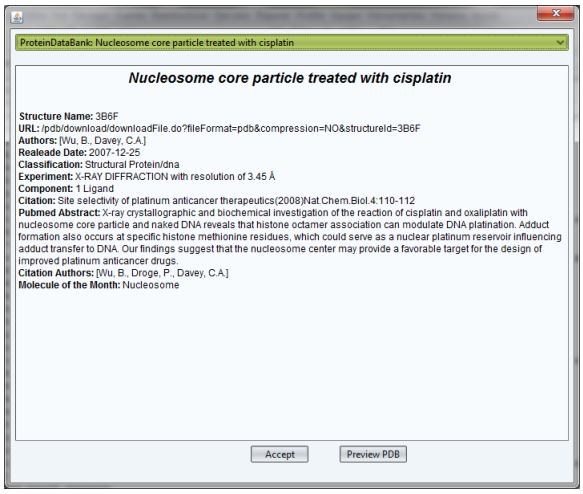


Figura 3.8: Informació sobre gens

i també l'usuari té la oportunitat de previsualitzar el fitxer ".pdb".

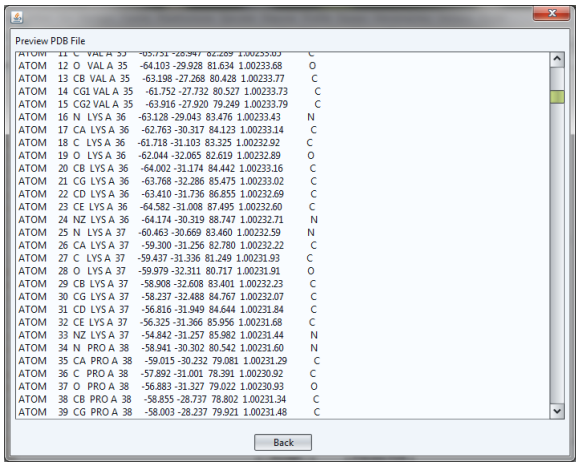


Figura 3.9: Previsualització del fitxer

Un cop s'ha decidit quins fitxers es volen descarregar es torna a la pantalla dels resultats per poder descarregar-los.

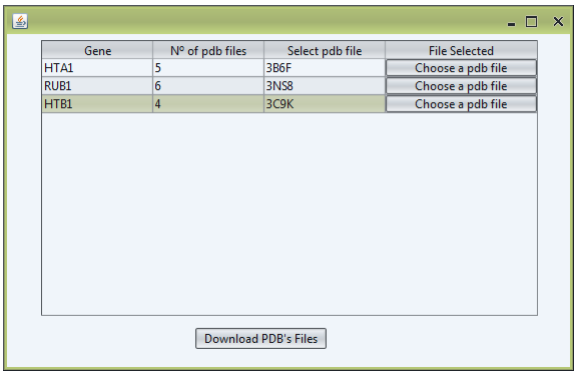


Figura 3.10: Descàrrega dels fitxers

3.3 Docking

Quan s'obtenen els fitxers ".pdb" s'inicia el procés de docking, on s'ha de fer parelles amb els fitxers obtinguts al pas anterior. Per fer les parelles primer es selecciona un gen del desplegable i després es selecciona un segon gen de la llista i s'afegeix aquesta parella per començar el procés de docking.

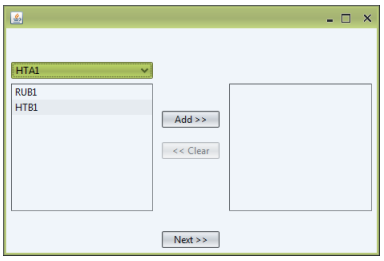


Figura 3.11: Parelles de fitxers per iniciar el docking

Després es configuren els paràmetres necessaris per poder iniciar el docking, per defecte els paràmetres necessaris són els següents, a l'Apèndix C s'explica el significat de cadascun d'ells:

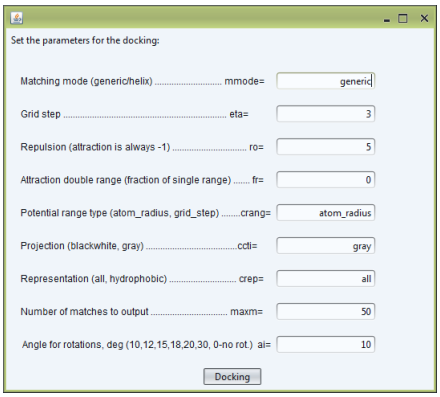


Figura 3.12: Paràmetres

El resultat final es retorna, també en format ".pdb", el qual si es desitja es pot visualitzar en mode gràfic gràcies al programa de visualització de PDB SwissPdbViewer. El resultat d'aquest acoblament molecular ha estat el següent:

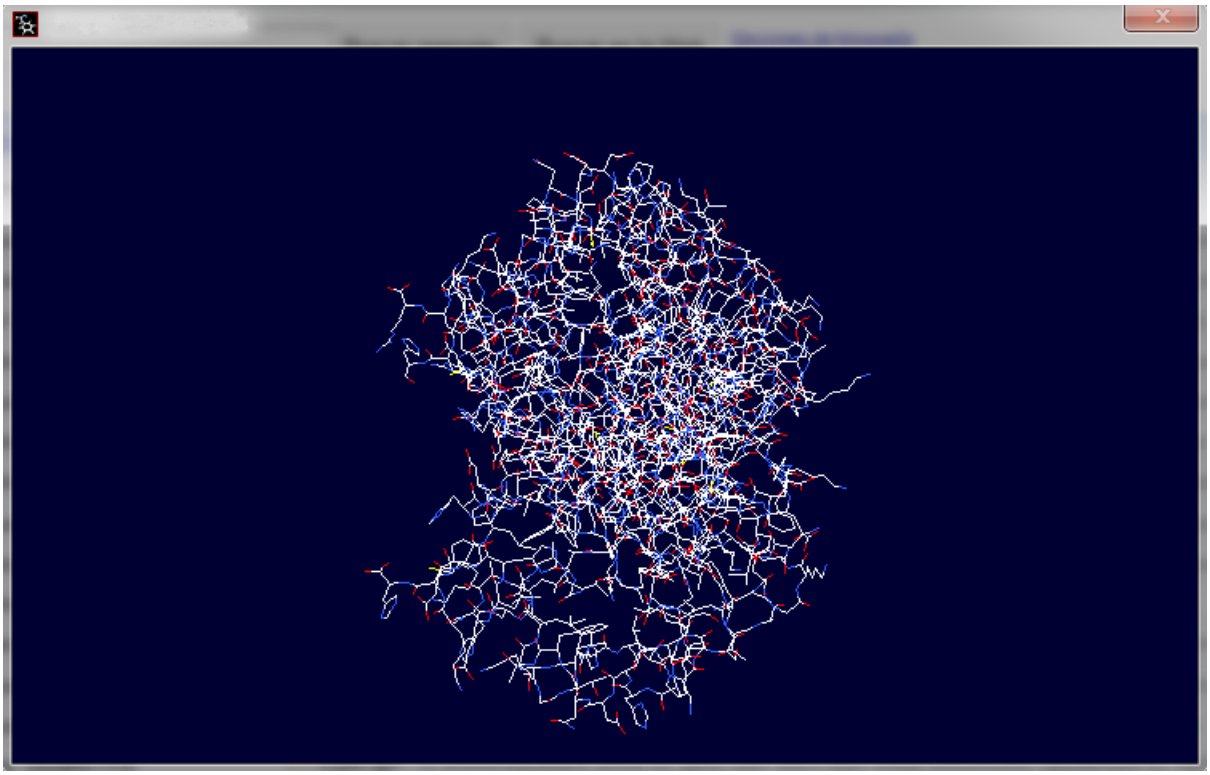
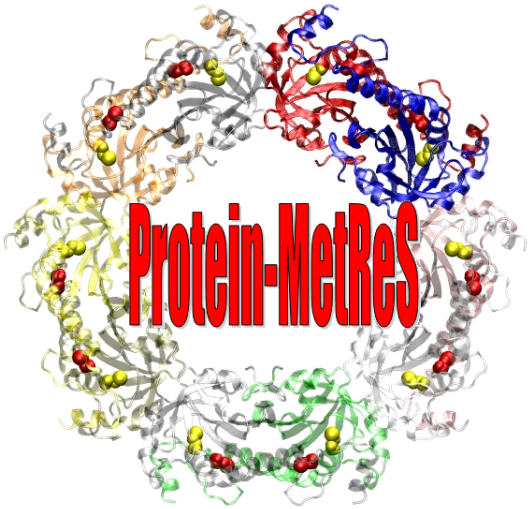


Figura 3.13: Visualització d'una proteïna

Capítol 4

Manual Protein-MetReS



4.1 Introduction

This is the user manual for Protein-MetReS. This manual will help you to understand all the functionalities of the applications and help you use the application appropriately.

Protein-MetReS is an application whose goal is to use the structure of your protein of interest, either by downloading it from the protein databank or the SwissModel repository or by creating model of that structure, for predictive protein docking in order to assist in the reconstruction of protein and gene co-occurrence networks of functional interactions. Combined with other data, these networks form the basis of an initial network reconstruction that will allow you to understand what does your protein or gene of interest do in your favorite organisms and whith which other proteins or genes does your protein work in order to achieve its appropriate biological function. Protein-MetReS relies on a central database with the genomes and gene annotation of more than 1000 organisms. It is this repository of gene names and functions that is accessed by the application when you

choose your organism of interest. Protein-Metres also relies on the structures contained in the Protein databank and in the SwissModel repository, and on the functionality of several local and internet servers for protein structure modeling and docking.

Protein-MetReS is conceived to be used by Molecular and Cellular Biologists. It can mine identify protein structures that are available online, at the PDB and the SwissModel repository. However, due to the non-standard nomenclature used to identify such structure, the program will sometimes identify incorrect structures. The user should carefully check the information provided by the program to filter these results. The program itself offers several control options that allow the user to do this. Similarly, when no structure is found, the user can create a model for the relevant protein, based on its sequence. Several servers that perform this function are available and the user can choose between them and filter the results in order to choose the best possible model. Finally, the user can also choose the appropriate parameters for the docking between the different structures.This manual will explain how these options work and which ones you should use, depending on what your goal is.

A couple of warnings are at hand. First, this is the first stable version of Protein-MetReS. We have tested extensively and it was widely and appropriately executed and tested by the overall research group. Although it was tested. However, we do not guarantee that the results are error free and you use it at your own responsibility. Second, given that Protein-MetReS searches the Internet and analyzes documents on the fly, you search and analysis might take a while. It may not look like it, but a very large amount of information is being analyzed as you run the program. As more people use Protein-MetReS, the process of search-analysis will become faster, because results from previous iterations of any given search-analysis will be stored locally and the program will only need to analyze new documents that are published since the last time a specific search was executed. Third, you need to have an active Internet connection to use Protein-MetReS. The application checks our local server to download gene information and previously compiled searches and uses the Internet to search for and download new documents.
In the next chapter we will start discussing the program.

4.2 Downloading Protein-MetReS and registering for usage

Protein-MetReS can be downloaded in this link (Figure 4.1). This application is written in JAVA and has been extensively tested in both Linux and Windows systems. Some tests was also performed in MAC operating systems.

Figura 4.1: Download page of Protein-MetReS

Once you downloaded the program, please create a directory in your computer where you will place the application (e. g. c://Program Files/Biblio_MetReS in Windows or /home/("user")/Desktop/ in Linux).
If you don't have user account, you will need to register. You can register in for use at this link (Figure 4.2) or you can run application and in the main menu you can click "Register" button. Once you click this button, application will redirect at the same page.

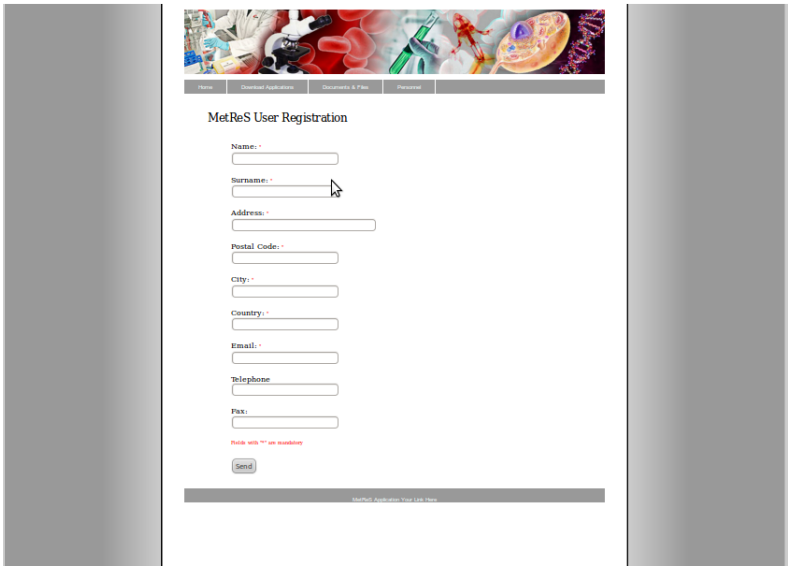


Figura 4.2: Protein-MetReS User Registration Web Page

Once you have registered, you will be receive an e-mail with your login name, your password, and an activation link. Clicking this link is essential for your user to become active.

4.3 Selecting your organism of interest

After you have registered and activated your acccount you can fully use Protein-MetReS. When you double click on the Protein-MetReS icon, a login windows opens. Please enter with your Username and Password.

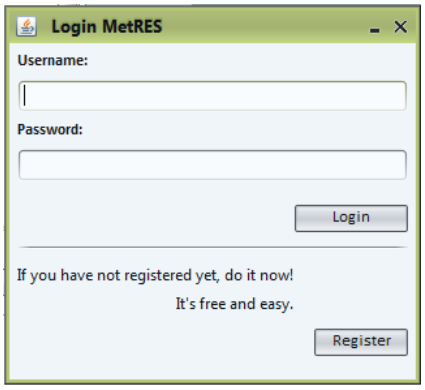


Figura 4.3: Login MetReS Frame

Once you're logged in, the Organism Selection window will be shown. Here you can select your organism of interest from a list of more than 1000 organisms with fully sequenced genomes that is available.

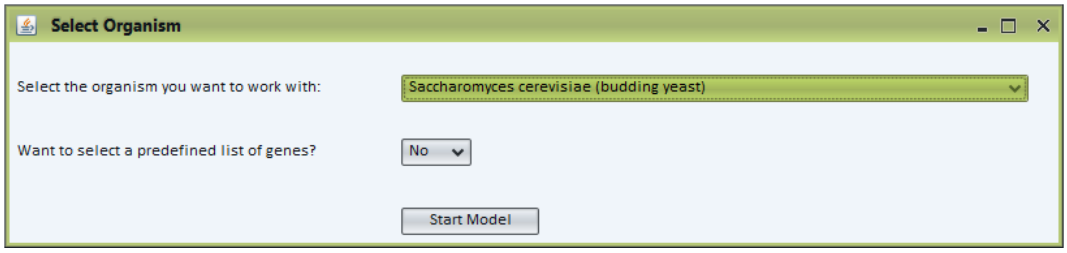


Figura 4.4: Select Organism Frame

To continue using the program you have two options. You can select a predefined list of genes or not. If you select "No" option, Protein-MetReS will open a windows with a list of genes related with selected organism.

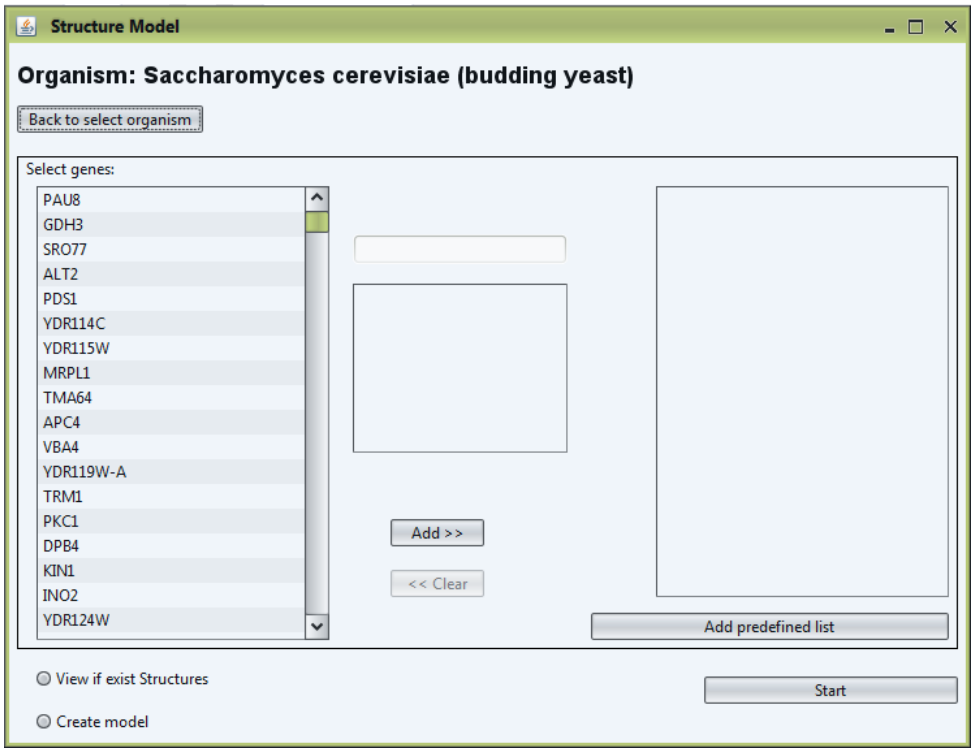


Figura 4.5: Structure Model Frame

You can choose gens of list and then you have two options, you need to select at least two gens if the application does not work.

4.3.1 View if exist Structure

If you select this option, you will search the Protein Data Bank and/or SwissModel repository in search of coincidences for structures(at the protein databank) or models of structures (SMR) for the gens you selected. These two servers provides a variety of tools and resources to search pre-existing genes. Once you selected options you start to search.

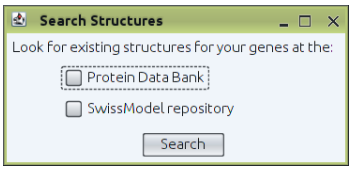


Figura 4.6: Look for existing structures Frame

Protein Data Bank archive is a repository of atomic coordinates and other information describing proteins and other important biological macromolecules. Structural biologists use methods such as X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy to determine the location of each atom relative to each other in the molecule. They then deposit this information, which is then annotated and publicly released into the archive by the wwPDB.

SwissModel repository is a database of annotated 3-dimensional comparative protein structure models generated by the fully automated homology-modelling pipeline SwissModel. The repository currently contains three-dimensional models for sequences from the UniProt knowledge base.

If the server find more than one result, the program creates a list for each gen with all results found with their respective information, so that once displayed you can download them.

4.3.2 Create Model

If no structures are found, then Protein-MetReS allows you to create homology models for your proteins of interest by sending their sequences to well known internet homology modelling servers. You can select between Swissmodel, 3DJIGSAW and Phyre. Locally, you can also choose to use Modeller or Rosetta.

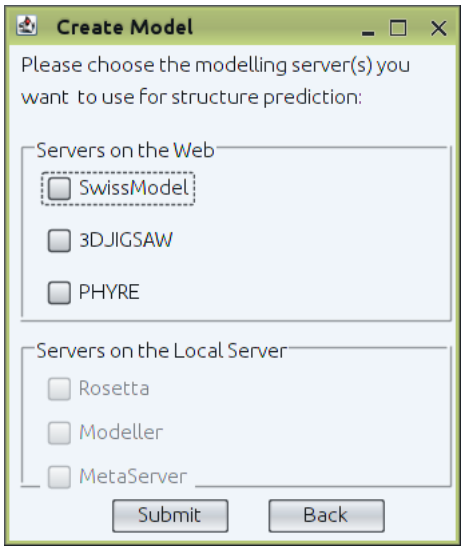


Figura 4.7: Create Model

SwissModel is a fully automated protein structure homology-modeling server. The purpose of this server is to make Protein Modelling accessible to all biochemists and molecular biologists WorldWide.

3DJIGSAW is an automated system to build three-dimensional models for proteins based on homologues of known structure. The program looks for homologous templates in our sequence databases (PFAM+PDB+nr) and splits the query sequence into domains.

Phyre is another protein structure prediction server. The functionaly is similar to 3DJIGSAW.

Once you select servers, the program sends a petitions of search genes to each servers. The results arrive to e-mail. When e-mails arrive, the program collects the information necessary to show, view and download files.

4.4 Returned Results

All results files are PDF files. PDB files list the atoms in each protein, and their 3D location in space. This file also includes a large "header" section of text that summarizes the protein, citation information, and the details of the structure solution, followed by the sequence and a long list of the atoms and their coordinates.

Both the structures found at the PDB, at the SMR and returned from the modeling servers are shown in the same intermediate results window. If there are more than one file for a given protein, the interface of the window allows you to read the information about the

different files, which in turn will help you decide which file contains the best structure. You should select this file to download. Figure 4.8 illustrates the flow of this window.

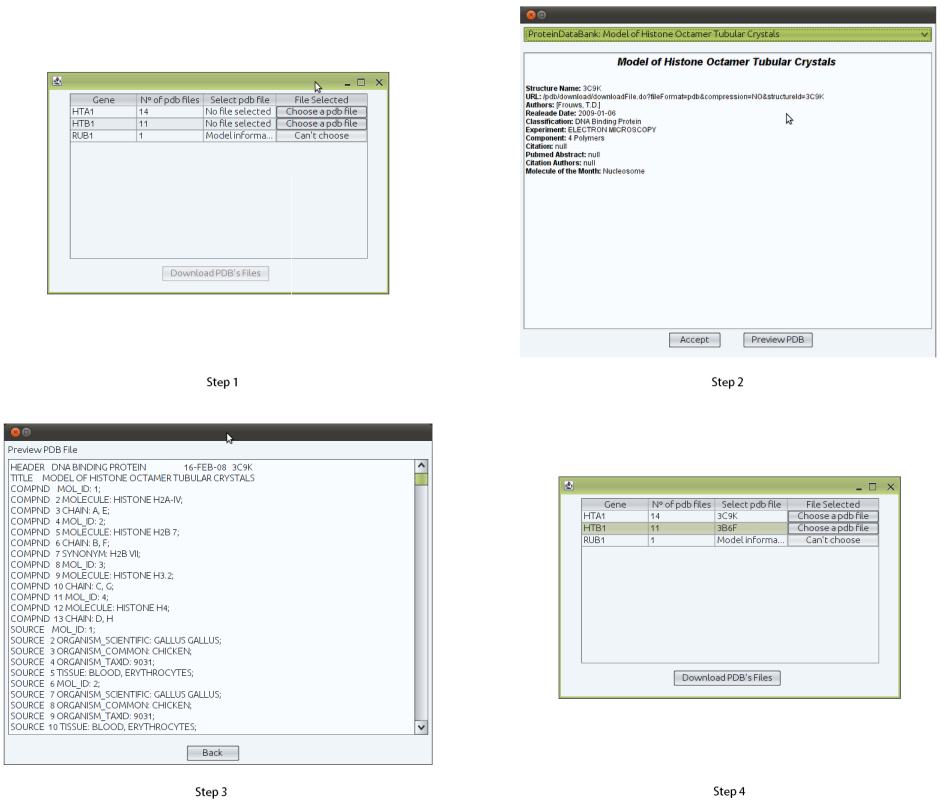


Figura 4.8: Treatment of results

4.5 Docking

After all PDB files are selected and downloaded to your local machine you obtain the files, program begins the Protein-MetReS allows you to start the process of docking between the different proteins.

Docking is done through another application called GRAMM (Global RAnge Molecular Matching). GRAMM is a program for protein docking. To predict the structure of a complex, it requires only the atomic coordinates of the two molecules. The program performs an exhaustive 6-dimensinal search through the relative translations and rotations of the molecules. The molecular pairs may be: two proteins, a protein and a smaller compound, two transmembrane helices, etc.

The GRAMM methodology is an empirical approach to smooting the intermolecular energy function by changing the range of the atom-atom potentials. The technique locales the area of the global minimum of intermolecular energy for structures of different accurancy. The quality of the prediction depends on the accurancy of the structures.

At the following window that will open, you can choose pairs of files for process of docking. You choose a gen of first list and a gen of second list, and then add to docking.

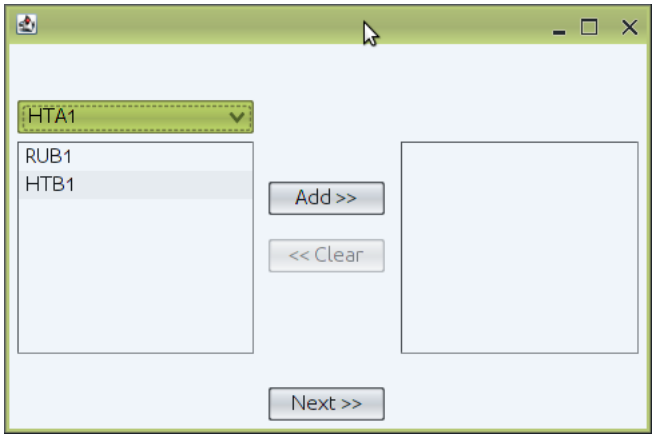


Figura 4.9: Docking process Step 1

The next and last step, you can modify the settings used in this process. The defaults settings are the most used.

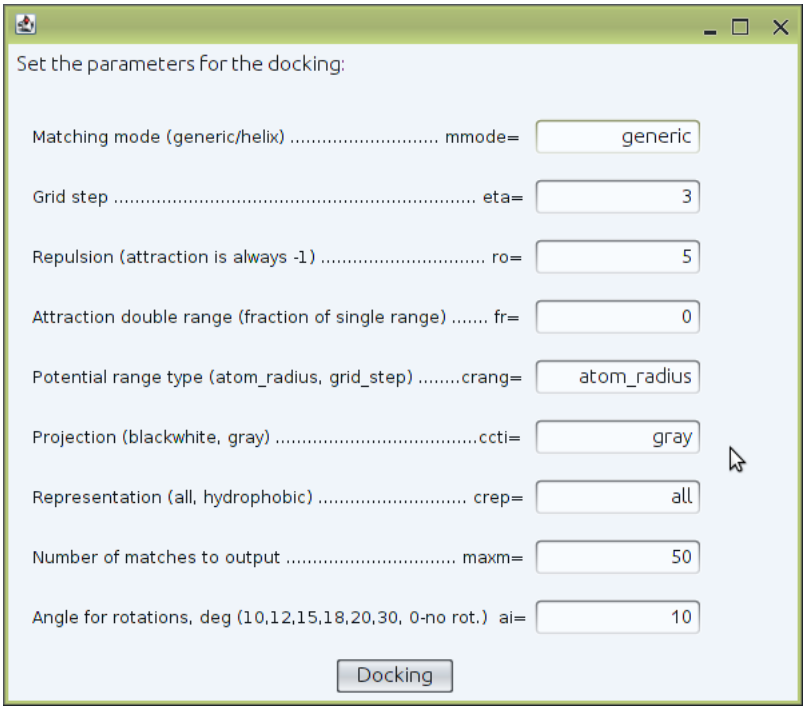


Figura 4.10: Docking process Step 2

The meaning of parameters are the follow:

- **mmode** Specifies the docking mode (**generic** or **helix**). In the generic mode, GRAMM tries all ligand's positions and orientations. In the helix mode, to save the computational time and to simplify the analysis of the results, GRAMM automatically discards configurations with large displacements along the helix axes and angles between helices larger. If you want to try all interhelical configurations, run GRAMM in the generic mode.
- **eta** Step of the grid (Katchalski-Katzir et al., 1992; Vakser, 1995,1996b); also the range of the atom-atom potential, in case of the 'gray' projection (Vakser, 1996a).
- **ro** Repulsion part of the potential, in arbitrary units (Vakser, 1996a).
- **fr** Attraction double range, mostly as an option for high-resolution docking Attraction double range, mostly as an option for high-resolution docking (Katchalski-Katzir et al., 1992; Vakser & Aflalo, 1994).
- **crang** Projection of an atom, as a sphere with the van der Waals radius (for high resolution docking) or the grid-step radius (for low resolution docking).
- **ccti** "yes-no" (blackwhite) or comulative (gray) projection /Vakser, 1995b, 1996b).
- **crep** Switch to the hydrophobic docking (Vakser & Aflalo, 1994).
- **maxm** Number of matches to output.
- **ai** Step fot the systematic search through the rotational coordinates.

When the process of docking ends, a message will be appear in your screen and the results will be in your system. These results are saved as PDB files that contain the structure of the docking complexes.

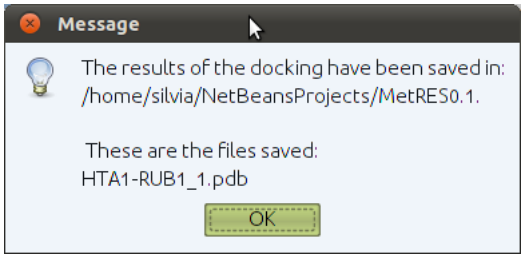


Figura 4.11: Final message

If you want to view these docking files, as well as the intermediate PDB files of the individual proteins, in graphic format you can use for example SwissViewer:

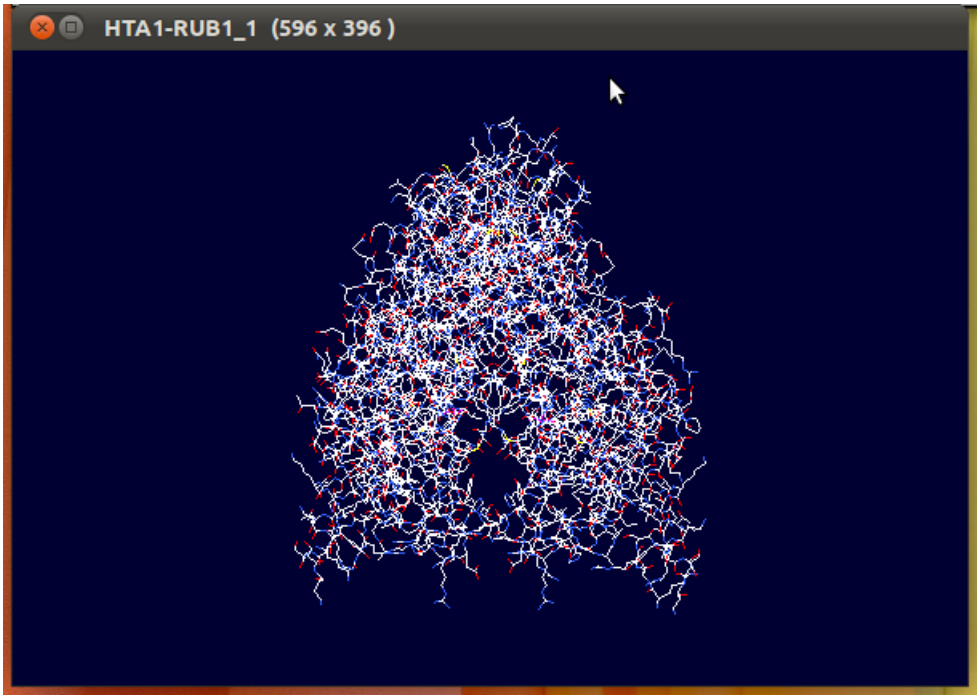


Figura 4.12: Returned graph

Capítol 5

Conclusions i treball futur

Un cop desenvolupat i implementat tot el que s'ha exposat en aquest document, podem dir que s'ha efectuat la creació d'una nova aplicació dins del marc del projecte MetReS i dins del camp de la biomedicina. Això no vol dir que l'aplicació hagi conclòs de la forma més òptima possible i donant la possibilitat a l'usuari d'explotar tots els recursos que ofereix l'aplicació, o millor dit, que en un futur oferirà. És per aquest motiu que a continuació es citaran alguns dels aspectes a millorar.

El primer punt important a millorar sobre l'aplicació és la optimització de la velocitat d'execució de les operacions, o més concretament, millorar la velocitat de resposta dels diferents servidors, ja que sofrir una demora de temps bastant elevada no és el més òptim, i més quan el propòsit de l'aplicació és treballar en un entorn professional.

Un altre aspecte a millorar per acabar de completar la funcionalitat de l'aplicació, seria la implementació dels servidors que actuen de forma local, ja que és un punt que no s'ha dut a terme per causa de la complexitat d'algun d'ells unida a la falta de temps útil.

Un altre punt important a millorar és integrar totes les aplicacions que puguin arribar a formar part del projecte MetReS i la realització del Benchmarking, que és una tècnica que s'utilitza per comprovar el rendiment del sistema.

Capítol 6

Referències

http://www.pdb.org/pdb/home/home.do

http://www.proteinmodelportal.org/?aid=queryModellingInteractiveBySeq&zid=async

http://bmm.cancerresearchuk.org/~3djigsaw/

http://www.sbg.bio.ic.ac.uk/~phyre/

http://www.accefyn.org.co/rasmol/PDB.htm

http://es.wikipedia.org/

http://novacripta.cbm.uam.es/bioweb

http://download.oracle.com/javase/tutorial/

http://www.encuentros.uma.es/encuentros87/prediccion.htm

`http://www.genome.jp/dbget-bin/www_bfind_sub?mode=bfind&max_hit=1000&locale=en&serv=kegg&dbkey=genes&keywords=FBA1&page=1`

Apèndix A

Fitxer ProteinDataBank.java

```
/*
    * To change this template, choose Tools | Templates
    * and open the template in the editor.
    */
    package Util.Structures;
    import Domain.SimpleLog;
    import java.io.BufferedReader;
    import java.io.IOException;
    import java.io.InputStreamReader;
    import java.net.URL;
    import java.util.ArrayList;
    import java.util.Collection;
    import java.util.List;
    import javax.swing.text.MutableAttributeSet;
    import javax.swing.text.html.HTML;
    import javax.swing.text.html.HTML.Attribute;
    import javax.swing.text.html.HTMLEditorKit;
    import javax.swing.text.html.parser.ParserDelegator;
    import Domain.Structures.InformationPDBank;
    import java.util.Iterator;

    public class ProteinDataBank extends HTMLEditorKit.ParserCallback{
      private SimpleLog logger;
      private String url;
      private boolean form=false;
      private boolean query=false;
      private int tagTable=0;
      private boolean queryTitle=false;
      private boolean textTitle1=false;
      String linkDownload;
      String title1;
      private boolean top=false;
      private boolean h4=false;
      private boolean textTitle2=false;
```

```
private boolean textTable1=false;
private int tagTable2=0;
String title2;
String encH4;
String table;
String tr1;
private boolean tablecontent=false;
private int tr=0;
private boolean trTag=false;
private boolean apartat4=false;
private boolean textAuthors=false;
private boolean textRelease=false;
private boolean textExperiment=false;
private boolean textCompound=false;
private boolean textCitation=false;
private boolean textMolecule=false;
private boolean td_key=false;
private boolean td_value=false;
private boolean textClassification=false;
private int trApartat4=0;
private int n=0;
private boolean pubmedAbstract=false;
private boolean tagAbstract=false;
private boolean citationAuthors=false;
private boolean textCitationAuthors=false;
private InformationPDBank info;
private List<InformationPDBank> listInfo= new ArrayList<InformationPDBank>();
private ArrayList<String> citationsAuthors;
private ArrayList<String> authors;
int nT=0, nC=0;
String citation;

public ProteinDataBank(SimpleLog logger, String url) {
  this.logger = logger;
  this.url=url;
}

public List<InformationPDBank> Execute() throws IOException{
  URL page = new URL(url);
 BufferedReader reader = new BufferedReader(new InputStreamReader(page.openStream()));
  try{
    new ParserDelegator().parse(reader, this, true);
  }
  catch (IOException e){
    logger.write("WARNING: Has not been able to obtain the information of ProteinData-
Bank.");
  }
```

```
    return listInfo;//retorna listInfo
  }
  public void handleComment(char[] data, int pos){}
  public void handleEndOfLineString(String eol){}

  public void handleEndTag(HTML.Tag tag, int pos){
    if(tag.equals(HTML.Tag.FORM))
    form=false;
    if(tag.equals(HTML.Tag.TABLE) && query && !apartat4){
      if(tagTable>0){
        tagTable-;
      }
      else{
        query=false;
      }
    }
    if(tag.equals(HTML.Tag.TD) && queryTitle){
      queryTitle=false;
    }
    if(queryTitle && tag.equals(HTML.Tag.A) && textTitle1){
      textTitle1=false;
    }
    if(tag.equals(HTML.Tag.TD) && top){
      top=false;
    }
    if(tag.equals(HTML.Tag.H4)){
      h4=false;
    }
    if(tag.equals(HTML.Tag.TR) && trTag && trApartat4==0){
      trTag=false;
      apartat4=false;
    }
    else if(tag.equals(HTML.Tag.TR) && trTag){
      if(trApartat4>2){
        trApartat4-;
      }
      else{
        trApartat4=0;
      }
    }
    if(tag.equals(HTML.Tag.TD) && td_key){
      td_key=false;
    }
    if(tag.equals(HTML.Tag.TD) && td_value && !apartat4){
      td_value=false;
      if(textAuthors)
        textAuthors=false;
      else if(textRelease)
```

```
            textRelease = false;
          else if(textExperiment)
            textExperiment = false;
          else if(textCompound)
            textCompound = false;
          else if(textCitation)
            textCitation = false;
          else if(textMolecule)
            textMolecule = false;
          else if(textClassification)
            textClassification = false;
        }
        if(tag.equals(HTML.Tag.DIV) && tagAbstract){
          tagAbstract=false;
        }
        if(tag.equals(HTML.Tag.SPAN) && textCitationAuthors){
          textCitationAuthors=false;
        }
      }
      public void handleError(String errorMsg, int pos){}
      public void handleMutableTag(HTML.Tag tag, MutableAttributeSet a, int pos){}
      public void handleSimpleTag(HTML.Tag tag, MutableAttributeSet a, int pos){}

     public void handleStartTag(HTML.Tag tag, MutableAttributeSet a, int pos){//IMPLEMENTAR
        if(tag.equals(HTML.Tag.FORM) && a.containsAttribute(Attribute.NAME, "Palffy")){
          form=true;
        }
        if(tag.equals(HTML.Tag.TABLE) && a.containsAttribute(Attribute.CLASS,"queryBlue")
&& form){
          info=new InformationPDBank();
          citationsAuthors = new ArrayList<String>();
          authors = new ArrayList<String>();
          nT=0;
          nC=0;
          n=0;
          citation=new String();
          tagTable++;
          query=true;
        }
       else if(tag.equals(HTML.Tag.TABLE) && a.containsAttribute(Attribute.CLASS,"queryWhite")
&& form){
          info=new InformationPDBank();
          citationsAuthors = new ArrayList<String>();
          authors = new ArrayList<String>();
          n=0;
          nT=0;
          nC=0;
          citation=new String();
```

```
          tagTable++;
          query=true;
        }
        else if(tag.equals(HTML.Tag.TABLE) && a.containsAttribute(Attribute.CLASS, "plain")
&& top){
          tagTable++;
          tablecontent=true;
          trApartat4=0;
          tr=0;
        }
        else if(tag.equals(HTML.Tag.TABLE) && trTag && !apartat4){
          tagTable++;
        }
        else if(tag.equals(HTML.Tag.TABLE) && query && !apartat4){
          tagTable++;
        }

        if(tag.equals(HTML.Tag.TD) && a.containsAttribute(Attribute.CLASS,"queryTitle") &&
query){
          queryTitle=true;
        }
       if(queryTitle && tag.equals(HTML.Tag.A) && a.containsAttribute(Attribute.CLASS,"qrb_structid")
){
          textTitle1=true;
        }
       if(queryTitle && tag.equals(HTML.Tag.A) && a.containsAttribute(Attribute.CLASS,"tooltip")){
          if(a.getAttribute(Attribute.HREF).toString().contains("download")){
            linkDownload=new String(a.getAttribute(Attribute.HREF).toString());
            info.setDownload(linkDownload);
          }
        }

       if(tag.equals(HTML.Tag.TD) && a.containsAttribute(Attribute.VALIGN,"top") && query
&& form){
          top=true;
        }
        if(tag.equals(HTML.Tag.H4) && top){
          h4=true;
        }
        if(tag.equals(HTML.Tag.A) && h4){
          textTitle2=true;
        }
        if(tag.equals(HTML.Tag.TR) && tablecontent && trApartat4==0){
          if(!apartat4){
            tr++;
            trTag=true;
          }
          else{
```

```
            trApartat4=1;
            trTag=true;
          }
        }
        else if(tag.equals(HTML.Tag.TR) && tablecontent){
          trApartat4++;
        }
        else if(tag.equals(HTML.Tag.TD) && a.containsAttribute(Attribute.CLASS, "se_value")
&& trTag){
          td_value=true;
          if(tr==1){
            textAuthors=true;
          }
          else if(tr==2){
            if(n==0){
              td_value=true;
              textRelease=true;
              n++;
            }
            else if(n == 1){
              td_value=true;
              textClassification=true;
              n=0;
              }
            }
            else if(tr==3){
              textExperiment=true;
            }
            else if(tr==4){
              textCompound=true;
              apartat4=true;
            }
            else if(tr==5){
              textCitation=true;
            }
            else if(tr==6){
              textMolecule=true;
            }
          }
        if(tag.equals(HTML.Tag.DIV) && a.containsAttribute(Attribute.CLASS,"abstract-large")
&& textCitation){
            tagAbstract=true;
          }
        if(tag.equals(HTML.Tag.SPAN) && a.containsAttribute(Attribute.CLASS, "qrb_subvalue")
&& textCitation){
            textCitationAuthors=true;
            tagAbstract=false;
          }
```

```
        }

      public void handleText(char[] data, int pos){
        if(textTitle1){
          title1=String.valueOf(data);
          info.setStructureName(title1);
        }
        if(h4){
          encH4=String.valueOf(data);
          info.setTitle(encH4);
        }
        if(textAuthors){
          String text=String.valueOf(data);
          String [] camps = text.split("\\n");
          for(int j=0; j<camps.length; j++){
            if(!camps[j].trim().startsWith(",")){
              authors.add(camps[j].trim());
            }
          }
          info.setAuthors(authors);
        }
        if(textRelease){
          String text2= String.valueOf(data);
          info.setRealeaseDate(text2);
        }
        if(textClassification){
          if(nT==0){
            nT++;
            String text2= String.valueOf(data);
            info.setClassification(text2);
          }
        }
        if(textExperiment){
          String text3= String.valueOf(data);
          info.setExperiment(text3);
        }
        if(textCompound && apartat4){
          String text4 = String.valueOf(data);
         if(!text4.trim().startsWith("[") && !text4.trim().startsWith("]") && !text4.trim().startsWith("|")
&& !text4.trim().startsWith("Hide") && !text4.trim().startsWith("Display")){
           if(text4.trim().endsWith("Polymer") || text4.trim().endsWith("Polymers") || text4.trim().endsWith("Li
|| text4.trim().endsWith("Ligans")){
              info.setComponent(text4);
            }
          }
        }
        if(textCitation && !tagAbstract && !textCitationAuthors){
          String text5 = String.valueOf(data);
```

```
        if(!text5.trim().startsWith("[") && !text5.trim().startsWith("]") && !text5.trim().startsWith("|")
&& !text5.trim().startsWith("Hide") && !text5.trim().startsWith("Display")){
            if(nC==0 || nC==1 || nC==2 || nC==4){
              nC++;
              citation=citation+text5.trim();
            }
            else if(nC==5){
              citation=citation+text5.trim();
              info.setCitation(citation);
            }
            else{
              nC++;
            }
          }
        }
        if(tagAbstract){
          String text6 = String.valueOf(data);
          String [] linies = text6.split("\\n");
          int i=0;
          while(i<linies.length){
           if(!linies[i].trim().contains("PubMed Abstract") && !linies[i].trim().contains("Citation
Authors")){
              info.setPubmedAbstract(linies[i]);
            }
            i++;
          }
        }
        if(textCitationAuthors){//Authors
          String text6 = String.valueOf(data);
          String linies[]= text6.split("\\n");
          for(int i=0; i<linies.length; i++){
            if(!linies[i].trim().startsWith(",")){
              String autor = linies[i].trim();
              citationsAuthors.add(autor);
            }
          }
          info.setCitationAuthors(citationsAuthors);
        }
        if(textMolecule){
          String text6 = String.valueOf(data);
          info.setMolecule(text6);
          listInfo.add(info);
        }
      }
      public static void main(String args[]) throws IOException {
       String url="http://www.pdb.org/pdb/results/results.do?outformat=&qrid=FEC99D0D&tabtoshow=Curre
        SimpleLog log= new SimpleLog();
        ProteinDataBank pdb=new ProteinDataBank(log,url);
```

```
        List<InformationPDBank> list=pdb.Execute();
    }
}
```

Apèndix B

Fitxer Mail.java

```
/*
    * To change this template, choose Tools | Templates
    * and open the template in the editor.
    */
    package Domain.Structures;

    import Domain.SimpleLog;
    import Util.Structures.SwissModelMail;
    import java.io.BufferedReader;
    import java.io.File;
    import java.io.FileNotFoundException;
    import java.io.FileOutputStream;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.InputStreamReader;
    import java.net.URL;
    import java.util.Date;
    import java.util.Properties;
    import javax.mail.Flags;
    import javax.mail.Folder;
    import javax.mail.Message;
    import javax.mail.MessagingException;
    import javax.mail.Multipart;
    import javax.mail.Part;
    import javax.mail.Session;
    import javax.mail.Store;

    public class Mail {
      private static String link_mail=null;
      private int resultsReceived;
      private static String gene=new String();
      private boolean areMails=false;
      private static String subject;
      private static String pathway;
```

```
private static SimpleLog logger;
public Mail(SimpleLog logger) {
  this.logger = logger;
}
public String ReadMail(int x, int resultsReceived){
  this.resultsReceived=resultsReceived;
  link_mail=null;
  String directori=new String();
  boolean download=false;
  boolean islink=false;
  boolean structure=false;
  String name_file=new String();
  pathway=System.getProperty("user.dir");
  File f=new File(pathway+"/FilesPDB/");
  if(!f.exists())
  f.mkdir();
  pathway=pathway+"/FilesPDB/";

  //We have to obtain the session mail
  Properties prop = new Properties();
  prop.setProperty("mail.imap.starttls.enable", "false");
  prop.setProperty("mail.imap.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
  prop.setProperty("mail.imap.socketFactory.fallback", "false");
  prop.setProperty("mail.imap.port", "993");
  prop.setProperty("mail.imap.socketFactory.port", "993");

  Session sesion = Session.getInstance(prop);
  try{
    //We obtain the folder of server
    Date d= new Date();
    switch(x){
    case 0:
      directori="3DJIGSAW";
      download=true;
      islink=false;
      structure=false;
      break;
    case 1:
      directori="PHYRE";
      download=false;
      islink=false;
      structure=false;
      break;
    case 2:
      directori="SWISSMODEL";
      download=false;
      islink=true;
      structure=false;
```

```
        break;
      case 3:
        directori="SWISSMODEL";
        download=false;
        islink=true;
        structure=true;
        break;
      }
      //We obtain the Store and the Folder to read mails.
      Store store = sesion.getStore("imap");
      store.connect("imap.gmail.com", "metres.cmb@gmail.com", "metres1234");
      Folder folder = store.getFolder(directori);
      folder.open(Folder.READ_ WRITE);
      //We obtain the messages
      Message[] message = folder.getMessages();
      //The subject value and the From value is showed
      for (int i = 0; i < message.length; i++){
        if(message[i].isSet(Flags.Flag.SEEN)==false){
          areMails=true;
          this.resultsReceived++;
          System.out.println("********** "+this.resultsReceived);
          subject=message[i].getSubject().trim();
          analyzeMessagePart(message[i],download,islink,name_file, structure);
          message[i].setFlag(Flags.Flag.SEEN, true);
          i=message.length;
        }
      }
      folder.close(false);
      store.close();
    }
    catch (Exception e){
      logger.write("SEVERE: Problems reading the e-mail account.");
    }
    if(link_ mail==null)
      return null;
    else{
      logger.write("INFO: mail received from "+directori);
      if(directori.equals("SWISSMODEL")){
        return directori+"**"+link_ mail+"**"+gene.trim()+"**"+this.resultsReceived;
      }
      else{
        return directori+"**"+link_ mail+"**"+subject+"**"+this.resultsReceived;
      }
    }
  }

  private static void analyzeMessagePart(Part part, boolean download, boolean islink, String
name_file, boolean structure){
```

```
try{
  if(part.isMimeType("multipart/*")){
  Multipart multi;
  multi = (Multipart) part.getContent();
  for(int j = 0; j < multi.getCount(); j++){
    analyzeMessagePart(multi.getBodyPart(j),download,islink, name_file, structure);
  }
}
else{
  if(part.isMimeType("text/*")){
    if(download==false && islink==false){
      name_file=pathway+subject+".pdb";
      guardaEnFichero(part, name_file);
    }
    else if(download==false && islink==true){
   BufferedReader in= new BufferedReader(new InputStreamReader(part.getInputStream()));
      String ent=in.readLine();
      String link_swiss= new String();
      while (ent != null){
        if((ent.startsWith("Check results at "))){
          link_swiss=ent.substring(17,182);
          link_mail=link_swiss;
          gene="gene";
          ent=null;
        }
        else{
          ent=in.readLine();
        }
      }
      //We open de url to check when the results is ended
      if(!link_swiss.isEmpty() && !structure){
        URL urlswiss= new URL(link_swiss);
        String link=null;
        while(link==null){
        SwissModelMail swiss=new SwissModelMail(new SimpleLog() ,urlswiss.toString());
          link=swiss.Execute();
        }
        String aux[]=link.split("\\*\\*");
        link_mail=aux[0];
        String a[]=aux[1].split("-");
        gene=a[0].trim();
      }
    }
    else{
      if(part.isMimeType("chemical/x-pdb")){
        name_file=pathway+subject+".pdb";
        guardaEnFichero(part, name_file);
      }
```

```
            }
          }
        }
      }
      catch (Exception e){
        logger.write("SEVERE: Problems reading the e-mail account.");
      }
    }

    private static void guardaEnFichero(Part unaParte, String name_file) throws FileNot-
FoundException, MessagingException, IOException{
      File f=new File(name_file);
      link_mail=f.getAbsolutePath();
      FileOutputStream fichero = new FileOutputStream(f);
      InputStream archivo = unaParte.getInputStream();
      byte[] bytes = new byte[1000];
      int leidos = 0;
      while ((leidos = archivo.read(bytes)) > 0){
        fichero.write(bytes, 0, leidos);
      }
    }
  }
```

Apèndix C

GRAMM. Fitxers de configuració

- Fitxer rpar.gr (paràmetres)

  Estableix els paràmetres del procés de docking. El valor d'un paràmetre ha d'aparèixer després del signe d'igualtat.

  mmode : Especifica el mode d'acoblament (genèric o d'hèlix). En el mode generic, GRAMM tracta totes les posicions i orientacions del ligand. En el mode hèlix descarta automàticament les configuracions amb un llarg desplaçament al llarg dels eixos de l'hèlix i angles entre les hèlixs més grans de l'indicat en l'arxiu rmol.gr (més endavant es pot veure).

  eta: Pas de la reixeta, calcula la posició adequanda entre el ligand i el receptor.

  ro: Part de repulsió del potencial, en unitats arbitraries.

  fr: Doble rang d'atracció. En la seva majoria com una opció d'alta resolució d'acoblament.

  crang: Projecció d'un àtom com una esfera amb el radi de Var Der Waals o el radi de pas de reixeta.

  ccti: Sí-no" (negre-blanc) o acumulativa (gris) projecció.

  crep: Canvi a docking hidrofòbic.

  maxm: Número de combinacions a la sortida.

ai: Pas per la cerca sistemàtica a través de les coordenades de rotació.

Exemple:

```
Matching mode (generic/helix) ..................... mmode= generic
Grid step ........................................... eta= 1.7
Repulsion (attraction is always -1) ................. ro= 5
Attraction double range (fraction of single range) ..... fr= 0
Potential range type (atom_radius, grid_step) ....... crang= atom_radius
Projection (blackwhite, gray) ..................... ..... ccti= gray
Representation (all, hydrophobic) ..................... crep= all
Number of matches to output ........................... maxm= 50
Angle for rotations, deg (10,12,15,18,20,30, 0-no rot.)  ai=10
```

Figura C.1: Format rpar.gr

- Fitxer rmol.gr (descripció de molècules)

  Les línies buides i les que comencen amb # són ignorades. Les dos primeres línies de l'exemple només serveixen per organitzar les dades. Dins del fitxer es poden introduir múltiples parelles moleculars (parella per línia). La primera molècula serà considerada com el "receptor" i la segona com el "ligand". Les dades es separen mitjançant la separació de l'espai.

  Filename: Arxiu amb les coordenades moleculars (PDB).

  Fragment:

  * - molècula complerta

  X - identificació de la cadena (majúscules i minúscules)

  xxxx-xxxx - número d'àtoms (primera passada)

  ID: Cadena de caràcters (sense espais entremig) per identificar cadascuna de les molècules. Aquests ID's seran utilitzats pel GRAMM com a nom dels arxius de sortida.

  parallel / antiparallel: Únicament mode hèlix.

  max. angle: Únicament mode hèlix. Estableix el límit per l'angle (en graus) entre els eixos principals.

  Exemple 1.

```
# Filename  Fragment  ID      Filename  Fragment  ID     [paral/anti  max.ang]
# ------------------------------------------------------------------------

pdb2hhb.ent 1-1069  2hhba   pdb2hhb.ent 1114-2256 2hhbb

pdb2ptc.ent    E    2ptce   pdb4pti.ent     *     4pti
```

Figura C.2: Exemple 1. Fitxer rmol.gr

Exemple 2.

```
# Filename  Fragment  ID      Filename  Fragment  ID     [paral/anti  max.ang]
# ------------------------------------------------------------------------

pdb1brd.ent  281-478  1brd2 pdb1brd.ent  554-785  1brd3   antipar       50

pdb1brd.ent  554-785  1brd3 pdb1brd.ent  822-961  1brd4   antipar       50
```

Figura C.3: Exemple 2. Fitxer rmol.gr

- Fitxer wlist.gr (llista de resultats)

  El format general és similar al del fitxer rmol.gr. Es poden especificar varies línies (una línia per cada arxiu de resultats). "File_of_predictions" és l'arxiu de sortida (resultats) del GRAMM. "Separate / Join" indica a l'aplicació com ha de construir els arxius individuals PDB per cada combinació o unir-los tots en un únic arxiu. "+init_lig" estableix l'opció per incloure les coordenades del ligand (abans del docking) dins del fitxer de resultats PDB.

  Exemple

```
# File_of_predictions   First_match   Last_match   separate/joint  +init_lig
# ------------------------------------------------------------------------

  1brd2-1brd3.res           1             10           joint           yes

  2hhba-2hhbb.res           3              7           separ           no
```

Figura C.4: Format wlist.gr